Collapsible pushdown automata and labeled recursion schemes. equivalence, safety and effective selection

Arnaud Carayol, Olivier Serre

Collapsible Pushdown Automata and Labeled Recursion Schemes Equivalence, Safety and Effective Selection

Arnaud Carayol
LIGM (Université Paris Est & CNRS), Paris, France

Olivier Serre
LIAFA (Université Paris Diderot – Paris 7 & CNRS), Paris,France

***Abstract*—Higher-order recursion schemes are rewriting systems for simply typed terms and they are known to be equi-expressive with collapsible pushdown automata (CPDA) for generating trees. We argue that CPDA are an essential model when working with recursion schemes. First, we give a new proof of the translation of schemes into CPDA that does not appeal to game semantics. Second, we show that this translation permits to revisit the safety constraint and allows CPDA to be seen as Krivine machines. Finally, we show that CPDA permit one to prove the effective MSO selection property for schemes, subsuming all known decidability results for MSO on schemes.**

***Keywords*-Recursion Schemes, Collapsible Pushdown Automata, Safety Constraint, MSO Effective Selection**

I. Introduction

Higher-order recursion schemes are rewriting systems for simply typed terms and in recent years they have received much attention as a method of constructing rich and robust classes of possibly infinite ranked trees. Remarkably these trees have decidable monadic second-order (MSO) theories, subsuming most of the examples of structures for which MSO is decidable. Since the original proof of Ong [15] based on traversals (a tool from game semantics), several alternative proofs (and extensions) were obtained using different techniques: automata [9], [2], intersection types [13], the Krivine machine [18].

In this article we focus on the automata approach. In [9], schemes were shown to be equi-expressive with an extension of the standard model of pushdown automata, called collapsible pushdown automata (CPDA). The translation from schemes into CPDA crucially relied on traversals. The decidability of MSO was obtained by solving parity games played on transition graphs of CPDA. In [2] a refinement of this proof was used to show that the family of trees generated by schemes is MSO-reflective, *i.e.* for any scheme $\mathcal{S}$ and any MSO formula $\varphi(x)$ with one first-order free variable x, one can build another scheme that produces the same tree as $\mathcal{S}$ except that now all nodes satisfying $\varphi(x)$ are marked.

In this article, we focus on the merits of CPDA for studying recursion schemes. As CPDA are more naturally associated with a labeled transition system (LTS) than with a tree, we introduce a variant of recursion schemes, *labeled recursion schemes*, that admit a canonical LTS. In both cases, the tree generated is simply the unfolding of the LTS. Although not technically difficult, we think that this notion and the associated family of LTS can be the subject of further studies.

Our first main result is a *simplified* and *syntactic* proof of the translation of a scheme into an equivalent CPDA. This is the first proof of the equi-expressivity result of [9] that *does not* use game semantics. A comparison of the obtained CPDA can be found at the beginning of Section III.

Furthermore this translation also permits one to view a CPDA as a Krivine machine, hence inheriting the simplified proof of [18] for decidability of μ-calculus model-checking.

We also show that when translating a *safe* scheme we obtain a CPDA that does not need to use the links. This result, independently obtained by Blum and Broadbent [1], unifies the work of [10] on safe schemes and sheds a new light on safety. As a spin-off result, we give a more natural definition of safety based on Damm's original work [6].

Finally, the true gain of the apparently more involved CPDA model is demonstrated by showing that the trees defined by recursion schemes enjoy the effective MSO selection property: for any scheme $\mathcal{S}$ and any formula $\exists X\, \varphi(X)$ if the tree t generated by $\mathcal{S}$ satisfies $\exists X\, \varphi(X)$, one can build another scheme generating the tree t where a set of nodes U satisfying $\varphi(X)$ is marked. This new result subsumes all previously known MSO-decidability results on recursion schemes (while keeping the same complexity, in particular the one of [13]) and relies on a careful analysis of the winning strategies in CPDA parity games.

II. Preliminaries

A. Trees and Terms

Let A be a finite alphabet. We denote by A^* the set of finite words over A. A tree t (with directions in A) is a non-empty prefix-closed subset of A^*. Elements of t are called *nodes* and ε is called the *root* of t. For any node $u \in t$ and any direction $a \in A$, we refer to ua, when it belongs to t, as the a-*child* of u. A node with no child is a *leaf*.

A *ranked alphabet* A is an alphabet that comes together with an arity function, $\varrho : A \to \mathbb{N}$. The *terms* built over a ranked alphabet A are those trees with directions $\overrightarrow{A} \stackrel{\text{def}}{=} \bigcup_{f\in A} \overrightarrow{f}$ where $\overrightarrow{f} = \{f_1, \ldots, f_{\varrho(f)}\}$ if $\varrho(f) > 0$ and $\overrightarrow{f} = \{f\}$ if $\varrho(f) = 0$. For a tree t with directions in $\overrightarrow{A}$ to be a

term, we require, for all nodes u, that the set $A_u = \{d \in \overrightarrow{A} \mid ud \in t\}$ is empty iff u ends with some $f \in A$ (hence $\varrho(f) = 0$) and if A_u is non-empty then it is equal to some $\overrightarrow{f} \in \overrightarrow{A}$. We denote by $\mathrm{Terms}(A)$ the set of terms over A.

For $c \in A$ of arity 0, we denote by c the term $\{\varepsilon, c\}$. For $f \in A$ of arity $n > 0$ and for terms $t_1, \ldots, t_n$, we denote by $f(t_1, \ldots, t_n)$ the term $\{\varepsilon\} \cup \bigcup_{i \in [1,n]} \{f_i\} \cdot t_i$. These notions are illustrated in Figure 1.

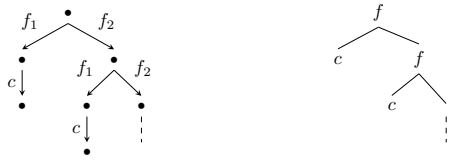


Figure 1. Two representations of the infinite term $f_2^*\{f_1 c, f_1, \varepsilon\} = f(c, f(c, f(\cdots)))$ over the ranked alphabet $\{f, c\}$ assuming that $\varrho(f) = 2$ and $\varrho(c) = 0$.

B. Labeled Transition Systems

A rooted labeled transition system (LTS for short) is an edge-labeled directed graph with a distinguished vertex, called the root. When considering LTS associated with computational models, it is usual to allow silent transitions. The symbol for silent transitions is usually ε but here, to avoid confusion with the empty word, we will instead use $\boldsymbol{e}$. We forbid a vertex to be the source of both a silent transition and of a non-silent transition. When Σ is an alphabet we let $\Sigma_{\boldsymbol{e}} = \Sigma \backslash \{\boldsymbol{e}\}$.

Formally, a *rooted labeled transition system with silent transitions* $\mathcal{L}$ is a tuple $\langle D, r, \Sigma, (\xrightarrow{a})_{a \in \Sigma} \rangle$ where D is a finite or countable set called the *domain*, $r \in D$ is a distinguished element called the *root*, Σ is a finite set of *labels* that contains a distinguished symbol denoted $\boldsymbol{e}$ and for all $a \in \Sigma$, $\xrightarrow{a} \subseteq D \times D$ is a binary relation on D.

For any $a \in \Sigma$ and any $(s, t) \in D^2$ we write $s \xrightarrow{a} t$ to indicate that $(s, t) \in \xrightarrow{a}$, and we refer to it as an *a-transition* with *source* s and *target* t. Moreover, we require that for all $s \in D$, if s is the source of a $\boldsymbol{e}$-transition, then s is not the source of any a-transition with $a \neq \boldsymbol{e}$. For a word $w = a_1 \cdots a_n \in \Sigma^*$, we define a binary relation $\xrightarrow{w}$ on D by letting $s \xrightarrow{w} t$ (meaning that $(s, t) \in \xrightarrow{w}$) if there exists a sequence $s_0, \ldots, s_n$ of elements in D such that $s_0 = s$, $s_n = t$, and for all $i \in [1, n]$, $s_{i-1} \xrightarrow{a_i} s_i$. These definitions are extended to languages over Σ by taking, for all $L \subseteq \Sigma^*$, the relation $\xrightarrow{L}$ to be the union of all $\xrightarrow{w}$ for $w \in L$.

For all words $w = a_1 \cdots a_n \in \Sigma_{\boldsymbol{e}}^*$, we denote by $\overset{w}{\Longrightarrow}$ the relation $\xrightarrow{L_w}$ where $L_w \stackrel{\text{def}}{=} \boldsymbol{e}^* a_1 \boldsymbol{e}^* \cdots \boldsymbol{e}^* a_n \boldsymbol{e}^*$ is the set of words over Σ obtained by inserting arbitrarily many occurrences of $\boldsymbol{e}$ in w.

An LTS is said to be *deterministic* if for all s, t_1 and t_2 in D and all a in Σ, if $s \xrightarrow{a} t_1$ and $s \xrightarrow{a} t_2$ then $t_1 = t_2$.

Caveat 1. *From now on, we always assume that the* LTS *we consider are deterministic.*

We associate a tree to every LTS $\mathcal{L}$, denoted $\mathrm{Tree}(\mathcal{L})$, with directions in $\Sigma_{\boldsymbol{e}}$, reflecting the possible behaviours of $\mathcal{L}$ starting from the root. For this we let $\mathrm{Tree}(\mathcal{L}) \stackrel{\text{def}}{=} \{w \in \Sigma_{\boldsymbol{e}}^* \mid \exists s \in D,\ r \overset{w}{\Longrightarrow} s\}$. As $\mathcal{L}$ is deterministic, $\mathrm{Tree}(\mathcal{L})$ is obtained by unfolding the underlying graph of $\mathcal{L}$ from its root and contracting all $\boldsymbol{e}$-transitions. Figure 2 presents an LTS with silent transitions together with its associated tree $\mathrm{Tree}(\mathcal{L})$.

As illustrated in Figure 2, the tree $\mathrm{Tree}(\mathcal{L})$ does not reflect the diverging behaviours of $\mathcal{L}$ (*i.e.* the ability to perform an infinite sequence of silent transitions). For instance in the LTS of Figure 2, the vertex s diverges whereas the vertex t does not. A more informative tree can be defined in which diverging behaviours are indicated by a $\bot$-child for some fresh symbol $\bot$. This tree, denoted $\mathrm{Tree}^{\bot}(\mathcal{L})$, is defined by letting $\mathrm{Tree}^{\bot}(\mathcal{L}) \stackrel{\text{def}}{=} \mathrm{Tree}(\mathcal{L}) \cup \{w\bot \in \Sigma_{\boldsymbol{e}}^* \bot \mid \forall n \geqslant 0, r \overset{w\boldsymbol{e}^n}{\Longrightarrow} s_n \text{ for some } s_n\}$.

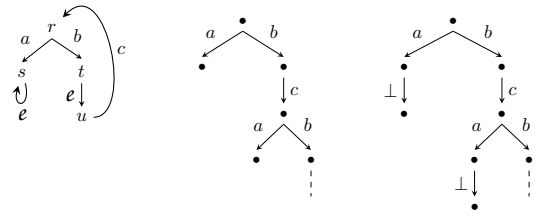


Figure 2. An LTS $\mathcal{L}$ with silent transitions of root r (on the left), the tree $\mathrm{Tree}(\mathcal{L})$ (in the center) and the tree $\mathrm{Tree}^{\bot}(\mathcal{L})$ (on the right).

C. Types, Applicative Terms

Types are generated by the grammar $\tau ::= o \mid \tau \to \tau$. Every type $\tau \neq o$ can be uniquely written as $\tau_1 \to (\tau_2 \to \cdots (\tau_n \to o) \ldots)$ where $n \geqslant 0$ and $\tau_1, \ldots, \tau_n$ are types. The number n is the *arity* of the type and is denoted by $\varrho(\tau)$. To simplify the notation, we take the convention that the arrow is associative to the right and we write $\tau_1 \to \cdots \to \tau_n \to o$ (or $(\tau_1, \ldots, \tau_n, o)$ to save space).

The *order* measures the nesting of a type: $ord(o) = 0$ and $ord(\tau_1 \to \tau_2) = \max(ord(\tau_1) + 1, ord(\tau_2))$.

Let X be a set of typed symbols. Every symbol $f \in X$ has associated a type τ; we write $f : \tau$ to mean that f has type τ. The set of *applicative terms of type τ generated from X*, denoted $\mathrm{Terms}_\tau(X)$, is defined by induction over the following rules. If $f : \tau$ is an element of X then $f \in \mathrm{Terms}_\tau(X)$; if $s \in \mathrm{Terms}_{\tau_1 \to \tau_2}(X)$ and $t \in \mathrm{Terms}_{\tau_1}(X)$ then the applicative term obtained by applying s to t, denoted $s\,t$, belongs to $\mathrm{Terms}_{\tau_2}(X)$. For every applicative term t, and every type τ, we write $t : \tau$ to mean that t is an applicative term of type τ. By convention, the application is considered to be left-associative, thus we write $t_1 t_2 t_3$ instead of $(t_1 t_2) t_3$.

Example 1. *Assuming that* $f : (\mathrm{o} \to \mathrm{o}) \to \mathrm{o} \to \mathrm{o}$, $g : \mathrm{o} \to \mathrm{o}$ *and* $c : \mathrm{o}$, *we have* $g\,c : \mathrm{o}$, $f\,g : \mathrm{o} \to \mathrm{o}$, $f\,g\,c = (f\,g)\,c : \mathrm{o}$ *and* $f\,(f\,g)\,c : \mathrm{o}$.

The set of subterms of t, denoted $\mathrm{Subs}(t)$, is inductively defined by $\mathrm{Subs}(f) = \{f\}$ for $f \in X$ and $\mathrm{Subs}(t_1\,t_2) = \mathrm{Subs}(t_1) \cup \mathrm{Subs}(t_2) \cup \{t_1\,t_2\}$. The subterms of the term $f\,(f\,g)\,c : \mathrm{o}$ in Example 1 are $f\,(f\,g)\,c$, f, $f\,g$, $f\,(f\,g)$, c and g. A less permissive notion is that of *argument subterms* of t, denoted $\mathrm{ASubs}(t)$, which only keep those subterms that appear as an argument. The set $\mathrm{ASubs}(t)$ is inductively defined by letting $\mathrm{ASubs}(t_1 t_2) = \mathrm{ASubs}(t_1) \cup \mathrm{ASubs}(t_2) \cup \{t_2\}$ and $\mathrm{ASubs}(f) = \varnothing$ for $f \in X$. In particular if $t = F t_1 \cdots t_n$, $\mathrm{ASubs}(t) = \cup_{i=1}^{n}(\mathrm{ASubs}(t_i) \cup \{t_i\})$. The argument subterms of $f\,(f\,g)\,c : \mathrm{o}$ are $f\,g$, c and g. In particular, for all terms t, one has $|\mathrm{ASubs}(t)| < |t|$ (the size $|t|$ of a term is the length of the word representation of t).

Remark 1. *A ranked alphabet A can be seen as a typed alphabet by assigning to every symbol f of A the type $\underbrace{o \to \cdots \to o}_{\varrho(f)} \to o$. In particular, every symbol in A has order 0 or 1. The finite terms over A (seen as a ranked alphabet) are in bijection with the applicative ground terms over A (seen as a typed alphabet).*

D. Labeled Recursion Schemes

Recursion schemes are grammars for simply typed terms, and they are often used to generate a possibly infinite term. Traditionally, recursion schemes are not associated with an LTS. Here we provide an alternative definition based on LTS.

For each type τ, we assume an infinite set V_τ of variables of type τ, such that V_{τ_1} and V_{τ_2} are disjoint whenever $\tau_1 \neq \tau_2$, and we write V for the union of those sets V_τ as τ ranges over types. We use letters $x, y, \varphi, \psi, \ldots$ to range over variables.

A deterministic *labeled recursion scheme* is a 5-tuple $\mathcal{S} = \langle\, \Sigma, N, \mathcal{R}, Z, \bot \,\rangle$ where

- Σ is a finite set of labels and $\bot$ is a distinguished symbol in Σ,
- N is a finite set of typed *non-terminals*; we use uppercase letters $F, G, H, \ldots$ to range over non-terminals,
- $Z : \mathrm{o} \in N$ is a distinguished *initial symbol* which does not appear in any right-hand side,
- $\mathcal{R}$ is a finite set of *production rules* of the form

$$F\,x_1 \cdots x_n \xrightarrow{a} e$$

where $a \in \Sigma \backslash \{\bot\}$, $F : (\tau_1, \cdots, \tau_n, o) \in N$, the x_is are distinct variables, each x_i is of type τ_i, and e is a ground term over $(N \backslash \{Z\}) \cup \{\, x_1, \ldots, x_n \,\}$.
In addition, we require that there is at most one production rule starting with a given non-terminal and labeled by a given symbol.

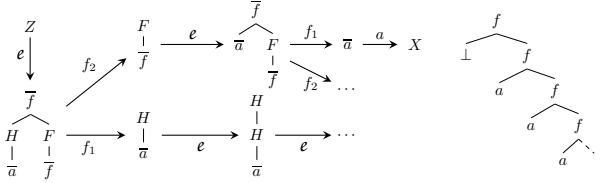


Figure 3. The LTS and the tree associated with the scheme $\mathcal{S}$ of Example 2.

The LTS associated with $\mathcal{S}$ has the set of ground terms over N as domain, the initial symbol Z as root, and, for all $a \in \Sigma$, the relation $\xrightarrow{a}$ is defined by:

$$F\,t_1 \ldots t_{\varrho(F)} \xrightarrow{a} e[t_1/x_1, \ldots, t_{\varrho(F)}/x_{\varrho(F)}]$$

if $F\,x_1 \cdots x_n \xrightarrow{a} e$ is a production rule.

The tree generated by a labeled recursion scheme $\mathcal{S}$, denoted $\mathrm{Tree}^{\bot}(\mathcal{S})$, is the tree $\mathrm{Tree}^{\bot}$ of its associated LTS. To use labeled recursion schemes to generate terms over ranked alphabet A, it is enough to enforce that for every non-terminal $F \in N$:

- either there is a unique production starting with F which is labeled by e,
- or there is a unique production starting with F which is labeled by some symbol c of arity 0 and whose right-hand side starts with a non-terminal that comes with no production rule in the scheme,
- or there exists a symbol $f \in A$ with $\varrho(f) > 0$ such that the set of labels of production rules starting with F is exactly $\overrightarrow{f}$.

Example 2. *Consider the order-*1 *scheme* $\mathcal{S} = \langle\, \Sigma, N, \mathcal{R}, Z, \bot \,\rangle$ *where* $\Sigma = \{a, f_1, f_2, \bot\}$, N *consists of* $Z, X, \overline{a} : o$, $H : (o, o)$, $\overline{f} : (o, o, o)$ *and* $F : ((o, o, o), o)$, *and* $\mathcal{R}$ *is given below*

$$\begin{array}{llll} Z & \xrightarrow{e} & \overline{f}\,(H\,\overline{a})\,(F\,\overline{f}) & \overline{a} \xrightarrow{a} X \\ H\,z & \xrightarrow{e} & H\,(H\,z) & \overline{f}\,x\,y \xrightarrow{f_1} x \\ F\,\varphi & \xrightarrow{e} & \varphi\,\overline{a}\,(F\,\varphi) & \overline{f}\,x\,y \xrightarrow{f_2} y \end{array}$$

The LTS *and the tree associated with $\mathcal{S}$ are depicted in Figure 3.*

Remark 2. *A more standard definition of* recursion schemes *[9] comes with a ranked alphabet A of* terminal symbols *that can be used in the right hand side of the rewriting rules; moreover the rules are no longer labeled. Applying rewriting rules from the initial symbol one derives finite terms over the set of terminal and non-terminal symbols. Replacing in such a term t any non-terminal, together with its argument, by a fresh symbol $\bot : \mathrm{o}$ leads a term $t^{\bot}$ over $A \cup \{\bot\}$. As the rewriting is confluent, there exists a supremum of all terms $t^{\bot}$ where t ranges over terms that can be rewritten from the initial symbol, and this (possibly infinite) term is defined as the* value term *of the scheme.*

It is easily seen that labeled recursion schemes and (usual) recursions schemes generate the same terms; the translations are linear and preserve both order and arity.

E. Examples of Trees Defined by Labeled Recursion Schemes

We provide some examples of trees defined by labeled recursion schemes. Given a language L over Σ, we denote by $\mathrm{Pref}(L)$ the tree containing all prefixes of words in L.

Example 3. *Using order-2 schemes, it is possible to go beyond deterministic context-free languages and to define for instance the tree* $T_1 = \mathrm{Pref}(\{a^n b^n c^n \mid n \geqslant 0\})$. *Consider for instance the order-2 scheme* $\mathcal{S}_1$ *given by:*

$$\begin{array}{llll} Z & \xrightarrow{a} & F\,I\,(\mathrm{K}\,C\,I) & B\,x \xrightarrow{b} x \\ F\,\varphi\,\psi & \xrightarrow{a} & F\,(\mathrm{K}\,B\,\varphi)\,(\mathrm{K}\,C\,\psi) & C\,x \xrightarrow{c} x \\ F\,\varphi\,\psi & \xrightarrow{b} & \varphi(\psi\,X) & I\,x \xrightarrow{e} x \\ \mathrm{K}\,\varphi\,\psi\,x & \xrightarrow{e} & \varphi(\psi(x)) & \end{array}$$

with $Z, X : o$, $B, C, I : o \to o$, $F : ((o \to o), (o \to o), o)$ *and* $\mathrm{K} : ((o \to o), (o \to o), o, o)$.

Intuitively, the non-terminal K *plays the role of the composition of functions of type* $o \to o$ (i.e. *for any terms* $F_1, F_2 : o \to o$ *and* $t : o$, $\mathrm{K}\,F_1\,F_2 t \xrightarrow{e} F_1(F_2\,t)$). *For any term* $G : o \to o$, *we define* G^n *for all* $n \geqslant 0$ *by taking* $G^0 = I$ *and* $G^{n+1} = \mathrm{K}\,G\,G^n$. *For any ground term* t, $G^n\,t$ *behaves as* $\underbrace{G\,(\ldots(G}_{n}(It))\ldots)$ *and in particular* $B^n\,X \overset{b^n}{\Longrightarrow} X$. *For all* $n \geqslant 0$, *we have:* $Z \xrightarrow{a^n} F\,B^{n-1}\,C^n \xrightarrow{b} B^{n-1}(C^n\,X) \overset{b^{n-1}c^n}{\Longrightarrow} X$.

Example 4. *We present a tree* T_U *proposed by Urzyczyn which exemplify the full expressivity of order-2 schemes (see Section IV). The tree* T_U *has directions in* $\{(,), \star\}$. *A word over* $\{(,)\}$ *is* well bracketed *if it has as many opening brackets as closing brackets and if for every prefix the number of opening brackets is not smaller than the number of closing brackets.*

The language U *is defined as the set of words of the form* $w\star^n$ *where* w *is a prefix of a well-bracketed word and* n *is equal to* $|w| - |u| + 1$ *where* u *is the longest suffix of* w *which is well-bracketed. In other words,* n *equals* 1 *if* w *is well-bracketed, and otherwise it is equal to the index of the last unmatched opening bracket plus one.*

For instance, the words $()((()) \star\star\star\star$ *and* $()()()\star$ *belong to* U. *The tree* T_U *is simply* $\mathrm{Pref}(U)$. *The following scheme* $\mathcal{S}_U$ *generates* T_U.

$$\begin{array}{llll} Z & \xrightarrow{e} & G\,(H\,X) & F\,\varphi\,x\,y \xrightarrow{(} F\,(F\varphi x)\,y\,(Hy) \\ G\,z & \xrightarrow{(} & F\,G\,z\,(Hz) & F\,\varphi\,x\,y \xrightarrow{)} \varphi\,(H\,y) \\ G\,z & \xrightarrow{\star} & X & F\,\varphi\,x\,y \xrightarrow{\star} x \\ H\,u & \xrightarrow{\star} & u & \end{array}$$

with $Z, X : o$, $G, H : o \to o$ *and* $F : (o \to o, o, o)$.

To better explain the inner workings of this scheme, let us introduce some syntactic sugar. With every integer, we associate a ground term by letting $\mathbf{0} = X$ *and, for all* $n \geqslant 0$, $\mathbf{n+1} = H\,\mathbf{n}$. *With every sequence* $[\mathbf{n_1} \ldots \mathbf{n_\ell}]$ *of integers, we associate a term of type* $o \to o$ *by letting* $[\,] = G$ *and* $[\mathbf{n_1} \ldots \mathbf{n_\ell}\,\mathbf{n_{\ell+1}}] = F\,[\mathbf{n_1} \ldots \mathbf{n_\ell}]\,\mathbf{n_{\ell+1}}$. *Finally we write* $([\mathbf{n_1} \ldots \mathbf{n_\ell}], \mathbf{n})$ *to denote the ground term* $[\mathbf{n_1} \ldots \mathbf{n_\ell}]\,\mathbf{n}$.

The scheme can be revisited as follows (note that the two rules labelled by (*are now merged):*

$$\begin{array}{llll} Z & \xrightarrow{e} & ([\,], \mathbf{1}) & ([\,], \mathbf{n}) \xrightarrow{\star} \mathbf{0} \\ ([\mathbf{n_1} \ldots \mathbf{n_\ell}], \mathbf{n}) & \xrightarrow{\star} & \mathbf{n_\ell} & \mathbf{n+1} \xrightarrow{\star} \mathbf{n} \\ ([\mathbf{n_1} \ldots \mathbf{n_\ell}], \mathbf{n}) & \xrightarrow{(} & ([\mathbf{n_1} \ldots \mathbf{n_\ell}\,\mathbf{n}], \mathbf{n+1}) & \\ ([\mathbf{n_1} \ldots \mathbf{n_\ell}], \mathbf{n}) & \xrightarrow{)} & ([\mathbf{n_1} \ldots \mathbf{n_{\ell-1}}], \mathbf{n+1}) & \end{array}$$

Let $w = w_0 \ldots w_{|w|-1}$ *be a prefix of a well-bracketed word. We have* $Z \overset{w}{\Longrightarrow} ([\,\mathbf{n_1} \ldots \mathbf{n_\ell}\,], |\mathbf{w}| + \mathbf{1})$ *where* $[\,n_1 \ldots n_\ell\,]$ *is the sequence (in increasing order) of those indices of unmatched opening brackets in* w. *In turn,* $([\,\mathbf{n_1} \ldots \mathbf{n_\ell}\,], |\mathbf{w}| + \mathbf{1}) \xrightarrow{\star} \mathbf{n_\ell} \xrightarrow{\star^{n_\ell}} \mathbf{0}$. *Hence, as expected, the number of* $\star$ *symbols is equal to* 1 *if* w *is well-bracketed (*i.e. $\ell = 0$*), and otherwise it is equal to the index of the last unmatched opening bracket plus one.*

F. Collapsible Pushdown Automata

Fix a finite stack alphabet Γ and a distinguished *bottom-of-stack symbol* $\perp \notin \Gamma$. An order-1 stack is a sequence $\perp, a_1, \ldots, a_\ell \in \perp\Gamma^*$ which is denoted $[\,\perp a_1 \ldots a_\ell\,]_1$. An *order-$k$ stack* (or a *k-stack*), for $k > 1$, is a non-empty sequence $s_1, \ldots, s_\ell$ of order-$(k-1)$ stacks which is written $[\,s_1 \ldots s_\ell\,]_k$. For convenience, we may sometimes see an element $a \in \Gamma$ as an order-0 stack, denoted $[\,a\,]_0$. We denote by Stacks_k the set of all order-k stacks and $\mathrm{Stacks} = \bigcup_{k \geqslant 1} \mathrm{Stacks}_k$ the set of all higher-order stacks. The height of the stack s denoted $|\,s\,|$ is simply the length of the sequence. We denote by $ord(s)$ the order of the stack s.

A *substack* of an order-1 stack $[\,\perp a_1 \ldots a_h\,]_1$ is a stack of the form $[\,\perp a_1 \ldots a_{h'}\,]_1$ for some $0 \leqslant h' \leqslant h$. A *substack* of an order-k stack $[\,s_1 \ldots s_h\,]_k$, for $k > 1$ is either a stack of the form $[\,s_1 \ldots s_{h'}\,]_k$ with $0 < h' \leqslant h$ or a stack of the form $[\,s_1 \ldots s_{h'}\,s'\,]_k$ with $0 \leqslant h' \leqslant h-1$ and s' a substack of $s_{h'+1}$. We denote by $s \sqsubseteq s'$ the fact that s is a substack of s'.

In addition to the operations $push_1^a$ and pop_1 that respectively pushes and pops a symbol in the topmost order-1 stack, one needs extra operations to deal with the higher-order stacks: the pop_k operation removes the topmost order-k stack, while the $push_k$ duplicates it.

For an order-n stack $s = [\,s_1 \ldots s_\ell\,]_n$ and an order-k stack t with $0 \leqslant k < n$, we define $s +\!\!+ t$ as the order-n stack obtained by pushing t on top of s:

$$s +\!\!+ t = \begin{cases} [\,s_1 \ldots s_\ell\,t\,]_n & \text{if } k = n-1, \\ [\,s_1 \ldots (s_\ell +\!\!+ t)\,]_n & \text{otherwise.} \end{cases}$$

We first define the (partial) operations pop_i and top_i with $i \geqslant 1$: $top_i(s)$ returns the top $(i-1)$-stack of s, and $pop_i(s)$ returns s with its top $(i-1)$-stack removed. Formally, for an order-n stack $[\, s_1 \cdots s_{\ell+1} \,]_n$ with $\ell \geqslant 0$

$$top_i(s) = \begin{cases} s_{\ell+1} & \text{if } i = n \\ top_i(s_{\ell+1}) & \text{if } i < n \end{cases}$$

$$pop_i(s) = \begin{cases} [\, s_1 \cdots s_\ell \,]_n & \text{if } i = n \text{ and } \ell \geqslant 1 \\ [\, s_1 \cdots s_\ell \, pop_i(s_{\ell+1}) \,] & \text{if } i < n \end{cases}$$

By abuse of notation, we let $top_{ord(s)+1}(s) = s$. Note that $pop_i(s)$ is defined if and only if the height of $top_{i+1}(s)$ is strictly greater than 1. For example $pop_2([\,[\,\bot\, a\, b\,]_1\,]_2)$ is undefined.

We now introduce the operations $push_i$ with $i \geqslant 2$ that duplicates the top $(i-1)$-stack of a given stack. More precisely, for an order-n stack s and for $2 \leqslant i \leqslant n$, we let $push_i(s) = s +\!\!+ top_i(s)$.

The last operation, $push_1^a$ pushes the symbol $a \in \Gamma$ on top of the top 1-stack. More precisely, for an order-n stack s and for a symbol $a \in \Gamma$, we let $push_1^a(s) = s +\!\!+ [\, a \,]_0$.

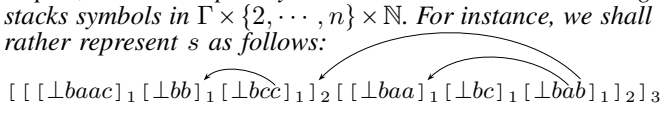
Example 5. *Let s be the order-3 stack of height 2 given by $s = [\,[\,[\,\bot baac\,]_1 [\,\bot bb\,]_1 [\,\bot bcc\,]_1 [\,\bot cba\,]_1\,]_2 [\,[\,\bot baa\,]_1 [\,\bot bc\,]_1 [\,\bot bab\,]_1\,]_2\,]_3$. Then $top_3(s)$ is the 2-stack $[\,[\,\bot baa\,]_1 [\,\bot bc\,]_1 [\,\bot bab\,]_1\,]_2$ and $pop_3(s)$ is the stack $s' = [\,[\,[\,\bot baac\,]_1 [\,\bot bb\,]_1 [\,\bot bcc\,]_1 [\,\bot cba\,]_1\,]_2\,]_3$. Note that $pop_3(pop_3(s))$ is undefined. Then $push_2(s')$ is the stack $[\,[\,[\,\bot baac\,]_1 [\,\bot bb\,]_1 [\,\bot bcc\,]_1 [\,\bot cba\,]_1 [\,\bot cba\,]_1\,]_2\,]_3$ and $push_1^c(s') = [\,[\,[\,\bot baac\,]_1 [\,\bot bb\,]_1 [\,\bot bcc\,]_1 [\,\bot cbac\,]_1\,]_2\,]_3$.*

We now define a richer structure of higher-order stacks where we allow links. Intuitively, a stack with links is a higher-order stack in which any symbol may have a link that points to an internal stack below it. This link may be used later to collapse part of the stack.

Order-n stacks with links are order-n stacks with a richer stack alphabet. Indeed, each symbol in the stack can be either an element $a \in \Gamma$ (*i.e.* not being the source of a link) or an element $(a, \ell, h) \in \Gamma \times \{2, \cdots, n\} \times \mathbb{N}$ (*i.e.* being the source of an ℓ-link pointing to the h-th $(\ell-1)$-stack inside the topmost ℓ-stack). Formally, order-n stacks with links over alphabet Γ are defined as order-n stacks [1] over alphabet $\Gamma \cup \Gamma \times \{2, \cdots, n\} \times \mathbb{N}$.

Example 6. *The stack s below is an order-3 stack with links $[\,[\,[\,\bot baac\,]_1 [\,\bot bb\,]_1 [\,\bot bc(c,2,2)\,]_1\,]_2 [\,[\,\bot baa\,]_1 [\,\bot bc\,]_1 [\,\bot b(a,2,1)(b,3,1)\,]_1\,]_2\,]_3$.*

To improve readability when displaying n-stacks in examples, we shall explicitly draw the links rather than using stacks symbols in $\Gamma \times \{2, \cdots, n\} \times \mathbb{N}$. For instance, we shall rather represent s as follows:



[1] Note that we therefore slightly generalise our previous definition as we implicitly use an infinite stack alphabet, but this does not introduce any technical change in the definition.

In addition to the previous operations pop_i, $push_i$ and $push_1^a$, we introduce two extra operations: one to create links, and the other to collapse the stack by following a link. Link creation is made when pushing a new stack symbol, and the target of an ℓ-link is always the $(\ell-1)$-stack below the topmost one. Formally, we define $push_1^{a,\ell}(s) = push_1^{(a,\ell,h)}$ where we let $h = |top_\ell(s)| - 1$ and require that $h > 1$.

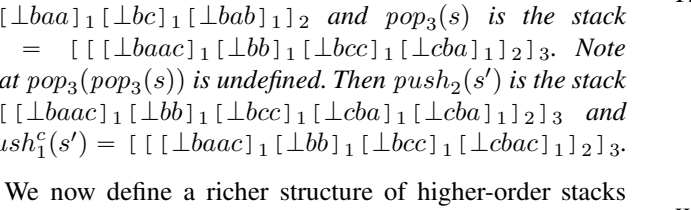
The collapse operation is defined only when the topmost symbol is the source of an ℓ-link, and results in truncating the topmost ℓ-stack to only keep the component below the target of the link. Formally, if $top_1(s) = (a, \ell, h)$ and $s = s' +\!\!+ [t_1 \cdots t_k]_\ell$ with $k > h$ we let $collapse(s) = s' +\!\!+ [t_1 \cdots t_h]_\ell$.

For any n, we let $\mathrm{Op}_n(\Gamma)$ denote the set of all operations over order-n stacks with links.

Example 7. *Let $s = [\,[\,[\,\bot a\,]_1\,]_2\ [\,[\,\bot\,]_1 [\,\bot a\,]_1\,]_2\,]_3$. We have*

$$\begin{aligned} push_1^{b,2}(s) &= [\,[\,[\,\bot a\,]_1\,]_2\ [\,[\,\bot\,]_1 [\,\bot a\, b\,]_1\,]_2\,]_3 \\ collapse\,(push_1^{b,2}(s)) &= [\,[\,[\,\bot a\,]_1\,]_2\ [\,[\,\bot\,]_1\,]_2\,]_3 \\ \underbrace{push_1^{c,3}(push_1^{b,2}(s))}_{\theta} &= [\,[\,[\,\bot a\,]_1\,]_2\ [\,[\,\bot\,]_1 [\,\bot a\, b\, c\,]_1\,]_2\,]_3. \end{aligned}$$

Then $push_2(\theta)$ and $push_3(\theta)$ are respectively



We have $collapse\,(push_2(\theta)) = collapse\,(push_3(\theta)) = collapse(\theta) = [\,[\,[\,\bot a\,]_1\,]_2\,]_3$.

An *order-n (deterministic) collapsible pushdown automaton* (n-CPDA) is a 5-tuple $\mathcal{A} = \langle\, \Sigma, \Gamma, Q, \delta, q_0 \,\rangle$ where Σ is an input alphabet containing a distinguished symbol denoted $\boldsymbol{e}$, Γ is a stack alphabet, Q is a finite set of control states, $q_0 \in Q$ is the initial state, and $\delta : Q \times (\Gamma \cup \{\bot\}) \times \Sigma \to Q \times \mathrm{Op}_n(\Gamma)$ is a (partial) transition function such that, for all $q \in Q$ and $\gamma \in \Gamma$, if $\delta(q, \gamma, \boldsymbol{e})$ is defined then for all $a \neq \boldsymbol{e}$, $\delta(q, \gamma, a)$ is undefined, *i.e.* if some $\boldsymbol{e}$-transition can be taken, then no other transition is possible. We require δ to respect the convention that $\bot$ cannot be pushed onto or popped from the stack.

Let $\mathcal{A} = \langle\, \Sigma, \Gamma, Q, \delta, q_0 \,\rangle$ be an n-CPDA. A *configuration* of an n-CPDA is a pair of the form (q, s) where $q \in Q$ and s is an n-stack with link over Γ; we call $(q_0, [[\cdots[\bot]_1 \cdots]_{n-1}]_n)$ the *initial configuration*. It is then natural to associate with $\mathcal{A}$ a deterministic LTS denoted $\mathcal{L}_\mathcal{A} = \langle\, D, r, \Sigma, (\xrightarrow{a})_{a \in \Sigma} \,\rangle$ and defined as follows. We let D be the set of all configurations of $\mathcal{A}$ and r be the initial one. Then for all $a \in \Sigma$ and all $(q, s), (q', s') \in D$ we have $(q, s) \xrightarrow{a} (q', s')$ if and only if $\delta(q, top_1(s), a) = (q', op)$ and $s' = op(s)$.

The tree generated by an n-CPDA $\mathcal{A}$, denoted $\mathrm{Tree}^{\perp}(\mathcal{A})$, is simply the tree $\mathrm{Tree}^{\perp}(\mathcal{L}_{\mathcal{A}})$ of its LTS.

III. From Recursion Schemes to Collapsible Pushdown Automata

In this section, we present a translation of schemes into CPDA. This translation generalizes at all orders the order-2 translation of [A4]. The translation from [9] assumes a normal form for the schemes but up to these normalisations, the CPDA obtained is the same as the one in [9]. Our contributions are to work direclty on schemes without normalisation and more importantly to prove the correctness of the translations without using game semantics as an intermediary tool as in [9]. Note that the converse translation from [9] (from CPDA into scheme) does not use game semantics and is therefore not presented here.

We construct, for any labeled recursion scheme $\mathcal{S}$, a collapsible pushdown automaton $\mathcal{A}$ of the same order defining the same tree as $\mathcal{S}$ – *i.e.* $\mathrm{Tree}^{\perp}(\mathcal{S}) = \mathrm{Tree}^{\perp}(\mathcal{A})$. To simplify the presentation, we assume that $\mathcal{S}$ does not contain any silent productions rule (*i.e.* production rule labeled by $\boldsymbol{e}$). If $\mathcal{S}$ were to contain silent transitions, we would treat the symbol $\boldsymbol{e}$ as any other symbol[2] in Σ. For the rest of this section, we fix a labeled recursion scheme $\langle\, \Sigma, N, \mathcal{R}, Z, \perp \,\rangle$ of order $n \geqslant 1$ without silent transitions.

The automaton $\mathcal{A}$ has a distinguished state, denoted $q_\star$, and with the configurations of the form $(q_\star, s)$ we will associate a ground term over N denoted by $[\![\, s \,]\!]$. Other configurations correspond to internal steps of the simulation and are only the source of silent transitions. To show that the two LTS define the same trees, we will establish that, for any reachable configuration of the form $(q_\star, s)$ and for any $a \in \Sigma$, the following holds:

- if $(q_\star, s) \xrightarrow[\mathcal{A}]{a\boldsymbol{e}^*} (q_\star, s')$ then $[\![\, s \,]\!] \xrightarrow[\mathcal{S}]{a} [\![\, s' \,]\!]$;
- if $[\![\, s \,]\!] \xrightarrow[\mathcal{S}]{a} t$ then $(q_\star, s) \xrightarrow[\mathcal{A}]{a\boldsymbol{e}^*} (q_\star, s')$ and $[\![\, s' \,]\!] = t$.

Hence, the main ingredient of the construction is the partial mapping $[\![\, \cdot \,]\!]$ associating with any order-n stack a ground term over N. The main difficulty is to guarantee that any rewriting rule of $\mathcal{S}$ applicable to the encoded term $[\![\, s \,]\!]$ can be simulated by applying a sequence of stack operations to s. In Section III-A, we present the mapping $[\![\, \cdot \,]\!]$ together with its basic properties; in Section III-B, we give the definition of $\mathcal{A}$ and prove the desired properties.

To simplify the presentation we assume, without loss of generality, that all productions starting with a non-terminal A have the same left-hand side (*i.e.* they use the same variables in the same order) and that two productions starting with different non-terminals do not share any variables. Hence a variable $x \in V$ appears in a unique left-hand side $A\, x_1 \ldots, x_{\varrho(A)}$ and we denote by $\mathrm{rk}(x)$ the index of x in the sequence $x_1 \cdots x_{\varrho(A)}$ (*i.e.* $x = x_{\mathrm{rk}(x)}$).

Throughout the whole section, we will illustrate definitions and constructions using as a running example the order-2 scheme $\mathcal{S}_U$ generating the tree T_U of Example 4.

A. Stacks Representing Terms.

The stack alphabet Γ consists of the initial symbol and of the right-hand sides of the rules in $\mathcal{R}$ and their argument subterms, *i.e.* $\Gamma \stackrel{\text{def}}{=} \{\, Z \,\} \cup \bigcup_{F\, x_1 \cdots x_{\varrho(x)} \xrightarrow{a} e} \{\, e \,\} \cup \mathrm{ASubs}(e)$.

For the scheme $\mathcal{S}_U$, one gets $\Gamma = \{x, y, z, u, \varphi\} \cup \{\, Z\,,\, G\,(H\,X)\,,\, H\,X\,,\, X\,,\, F\,(F\,\varphi\,x)\,y\,(Hy)\,,\, F\,\varphi\,x\,,\, Hy\,,\, F\,G\,z\,(Hz)\,,\, G\,,\, Hz\,,\, \varphi\,(Hy) \}$.

Notation 1. *For $\varphi \in V \cup N$, a φ-stack designates a stack whose top symbol starts with φ. By extension a stack s is said to be an N-stack (resp. a V-stack) if it is a φ-stack for some $\varphi \in N$ (resp. $\varphi \in V$).*

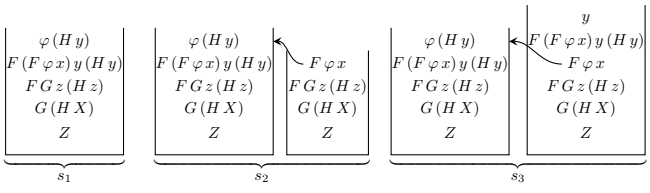
In order to represent a term in $\mathrm{Terms}(N)$, a stack over Γ must be *well-formed*, *i.e.* it must satisfy some syntactic conditions.

Definition 1 (Well-formed stack)**.** *A non-empty stack of order-n over Γ is* well-formed *if every non-empty substack r of s satisfies the following two conditions:*

- *if $top_1(r)$ is not equal to Z nor to $\perp$ then $pop_1(r)$ is an A-stack for some $A \in N$ and $top_1(r)$ belongs to an A-production rule,*
- *if $top_1(r)$ is of type τ of order $k > 0$ then $top_1(r)$ is the source of an $(n-k+1)$-link and $collapse(r)$ is a φ-stack for some variable $\varphi \in V$ of type τ.*

We denote by WStacks *the set of all well-formed stacks.*

Example 8. *For the scheme $\mathcal{S}_U$, the following order-2 stacks are well-formed.*



Notation 2. *We write $s :: t$ for $s \in$ WStacks and $t \in \Gamma$ to mean that if t belongs to the r.h.s. of a production starting with $A \in N$ then s is an A-stack. In particular, if $s \in$ WStacks then $pop_1(s) :: top_1(s)$. We denote by* CStacks *the set of such $s :: t$, and define the* size *of an element $s :: t$ as the pair $(|s|, |t|)$ where $|s|$ denotes the number of stack symbols in s and $|t|$ the length of the term t. When comparing sizes, we use the standard lexicographic (total) order over $\mathbb{N} \times \mathbb{N}$.*

In Definition 4, we will associate, with any well-formed stack s, a ground term over N that we refer to as the

[2]Formally, one labels all silent production rules of $\mathcal{S}$ by a fresh symbol e to obtain a labeled scheme $\mathcal{S}'$ without silent transitions. The construction presented in this section produces an automaton $\mathcal{A}'$ such that $\mathrm{Tree}^{\perp}(\mathcal{S}') = \mathrm{Tree}^{\perp}(\mathcal{A}')$. The automaton $\mathcal{A}$ obtained by replacing all e-labeled rules of $\mathcal{A}$ by $\boldsymbol{e}$ is such that $\mathrm{Tree}^{\perp}(\mathcal{S}) = \mathrm{Tree}^{\perp}(\mathcal{A})$.

value of s. To define this value, we first associate, with any element $s :: t$ in CStacks, a value denoted $[\![s :: t]\!]$. This value is a term over N of the same type as t. Intuitively, it is obtained by replacing the variables appearing in the term t by values encoded in the stack s, and one should therefore understand $[\![s :: t]\!]$ as the value of the term t in the context (or environment) of s. See Remark 3 below for natural connections with Krivine machine.

Definition 2. *For all $\varphi \in V \cup N$, all $k \in [1, \varrho(\varphi)]$ and all φ-stack $s \in$ WStacks, we define an element of* CStacks*, denoted $Arg_k(s)$, representing the k-th argument of the term represented by s. More precisely if the top symbol of s is $\varphi\, t_1 \cdots t_\ell$, we take:*

$$\begin{cases} Arg_k(s) & = & pop_1(s) :: t_k & \text{if } k \leqslant \ell, \\ Arg_k(s) & = & Arg_{k-\ell}(collapse(s)) & \text{otherwise.} \end{cases}$$

Definition 3. *For all $s :: t \in$ CStacks, we define the value of t in the context of s:*

$$\begin{cases} [\![s :: t_1 t_2]\!] & = & [\![s :: t_1]\!][\![s :: t_2]\!] & \text{if } t_1, t_2 \in \Gamma \\ [\![s :: A]\!] & = & A & \text{if } A \in N \\ [\![s :: x]\!] & = & [\![Arg_{\mathrm{rk}(x)}(s)]\!] & \text{if } x \in V \end{cases}$$

Let us provide some intuitions regarding the definition of $[\![s :: t]\!]$. Unsurprisingly $[\![s :: t]\!]$ is defined by structural induction on t, and the cases for the application and the non-terminal symbols are straightforward. It remains to consider the case where t is a variable x appearing in $\mathrm{rk}(x)$-th position in the left-hand side $A\, x_1 \cdots x_{\varrho(A)}$. As $s :: t \in$ CStacks, $top_1(s)$ is of the form $A t_1 \dots t_\ell$ for some $\ell \leqslant \varrho(A)$. Note that ℓ is not necessarily equal to $\varrho(A)$ meaning that some arguments of A might be missing. There are now two cases — that correspond to the two cases in the definition of $\mathrm{Arg}_k(s)$ — depending on whether x references to one of the t_i's (*i.e.* $\mathrm{rk}(x) \leqslant \ell$) or one of the missing arguments (*i.e.* $\mathrm{rk}(x) > \ell$):

- If $\mathrm{rk}(x) \leqslant \ell$ then the term associated with x in s is equal to the term associated with $t_{\mathrm{rk}(x)}$ in $pop_1(s)$, *i.e.* $[\![s :: x]\!] = [\![pop_1(s) :: t_{\mathrm{rk}(x)}]\!]$.
- If $\mathrm{rk}(x) > \ell$ then the term $[\![s :: x]\!]$ is obtained by following the link attached to $top_1(s)$. Recall that, as s is a well-formed stack and $top_1(s)$ is not of ground type (as $\ell < \varrho(A)$), there exists a link attached to $top_1(s)$. Moreover, $collapse(s)$, the stack obtained by following the link, has a top-symbol of the form $\varphi\, t'_1 \dots t'_m$ for some $\varphi \in V$ and $m \geqslant 0$. Intuitively, t'_i corresponds to the $(\ell + i)$-th argument of A. If $\mathrm{rk}(x)$ belongs to $[\ell + 1, \ell + m]$ then the term $[\![s :: x]\!]$ is defined to be the term $[\![pop_1(collapse(s)) :: t'_{\mathrm{rk}(x)-\ell}]\!]$. If $\mathrm{rk}(x)$ is greater than $\ell + m$ then the link attached to the top symbol of $collapse(s)$ is followed and the process is reiterated. As the size of the stack strictly decreases at each step this process terminates.

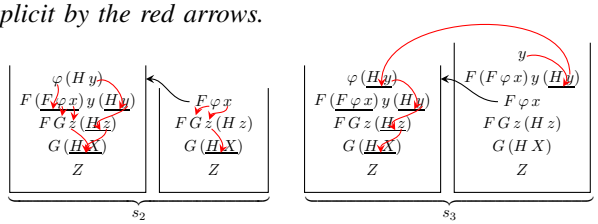
Now, if s is a well-formed φ-stack, its value is obtained by applying the value of φ in the context of $pop_1(s)$ to the value of all its $\varrho(\varphi)$ arguments. This leads to the following formal definition.

Definition 4. *The term associated with a well-formed φ-stack $s \in$ Stacks with $\varphi \in N \cup V$ is*

$$[\![s]\!] \stackrel{\text{def}}{=} [\![pop_1(s) :: \varphi]\!][\![Arg_1(s)]\!] \cdots [\![Arg_{\varrho(\varphi)}(s)]\!].$$

Equiv., if $top_1(s) : o$ then: $[\![s]\!] = [\![pop_1(s) :: top_1(s)]\!]$. If $top_1(s) : \tau_1 \to \ldots \to \tau_\ell \to \mathrm{o}$ then: $[\![s]\!] = [\![pop_1(s) :: top_1(s)]\!] [\![\mathrm{Arg}_1(collapse(s))]\!] \cdots [\![\mathrm{Arg}_\ell(collapse(s))]\!]$.

Example 9. *Let us consider the well-formed stacks s_2 and s_3 presented in Example 8. In the representation below the association between variables and their "values" are made explicit by the red arrows.*



$$[\![s_1]\!] = [\![s_2]\!] = F\, G\, (H\, X)\, (H(H(H(H\, X))))$$
$$[\![s_3]\!] = H(H(H(H(H\, X))))$$

The following lemma states the basic properties of the encoding $[\![\cdot]\!]$ and $\mathrm{Arg}_k(\cdot)$.

Lemma 1. *We have the following properties:*

1) *For all φ-stacks $s \in$ WStacks with $\varphi \in V \cup N$ of type $\tau_1 \to \ldots \to \tau_{\varrho(\varphi)} \to \mathrm{o}$ and for all $k \in [1, \varrho(\varphi)]$, $Arg_k(s)$ is equal to some $r :: t \in$ CStacks with t of type τ_k.*
2) *For all $s :: t \in$ CStacks with $t : \tau \in \Gamma$, $[\![s :: t]\!]$ is a term in $\mathrm{Terms}_\tau(N)$.*
3) *For all $s \in$ WStacks, $[\![s]\!]$ belongs to $\mathrm{Terms}_\mathrm{o}(N)$.*

We conclude with two fundamental properties of $\mathrm{Arg}_k(\cdot)$ that will allow us to simulate the rewriting of the scheme using stack operations and finite memory.

The first property is that the arguments represented by a well-formed stack are not modified when performing a $push_k$ operation. More precisely, for all φ-stacks $s \in$ WStacks with $\varphi \in N \cup V$, $[\![\mathrm{Arg}_\ell(push_k(s))]\!] = [\![\mathrm{Arg}_\ell(s)]\!]$ for all $\ell \in [1, \varrho(\varphi)]$ and all $k \in [2, m]$. This follows (by letting $r = top_k(s)$) from the following slightly more general result.

Lemma 2. *Let $k \in [2, m]$ and let $s = s' +\!\!+\, top_k(s) \in$ WStacks. For all non-empty φ-stacks $r \sqsubseteq top_k(s)$, $[\![Arg_\ell(s' +\!\!+ r)]\!] = [\![Arg_\ell(s +\!\!+ r)]\!]$ for all $\ell \in [1, \varrho(\varphi)]$.*

The next property will later be used to prove that any rewriting step can be simulated by a *finite* number of transitions in the automaton.

Lemma 3. *Let s be a φ-stack in* WStacks *for some $\varphi : \tau_1 \to \ldots \to \tau_{\varrho(\varphi)} \to o$ in $V \cup N$ and let $\ell \in [1, \varrho(\varphi)]$ with τ_ℓ of order $k > 0$. If $Arg_\ell(s)$ is equal to $r :: t \in$ CStacks with t starting with $\psi \in N \cup V$ then $pop_{n-k+1}(s) = pop_{n-k+1}(r)$, $|\, top_{n-k+1}(s)\,| > |\, top_{n-k+1}(r)\,|$.*

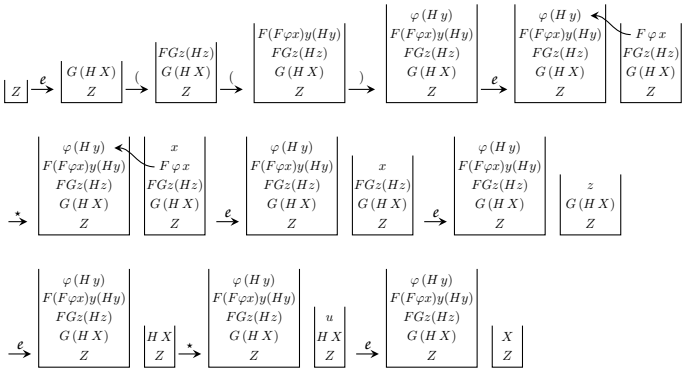
B. Simulating the LTS *of $\mathcal{S}$ on Stacks*

As an intermediate step, we define an LTS $\mathcal{M}$ over well-formed stacks and we prove that it generates the same tree as $\mathcal{S}$ (*i.e.* $\mathrm{Tree}^{\perp}(\mathcal{M}) = \mathrm{Tree}^{\perp}(\mathcal{S})$). From $\mathcal{M}$, a CPDA generating $\mathrm{Tree}^{\perp}(\mathcal{M})$ is then defined at the end of this section.

We let $\mathcal{M} = \langle\, \mathrm{WStacks}, [\ldots[\,\perp Z\,]\ldots]_n, \Sigma, (\xrightarrow[\mathcal{M}]{a})_{a\in\Sigma} \,\rangle$ and define the transitions as follows

- $s \xrightarrow[\mathcal{M}]{a} push_1^t(s)$ if s is an A-stack with $A \in N$ and $A\, x_1 \cdots x_{\varrho(A)} \xrightarrow{a} t \in \mathcal{R}$,
- $s \xrightarrow[\mathcal{M}]{e} push_1^t(r)$ if s is a φ-stack with $\varphi : o \in V$ and $\mathrm{Arg}_{\mathrm{rk}(\varphi)}(pop_1(s)) = r :: t$,
- $s \xrightarrow[\mathcal{M}]{e} push_1^{t,n-k+1}(r)$ if s is a φ-stack with $\varphi : \tau \in V$ of order $k > 0$ and $\mathrm{Arg}_{\mathrm{rk}(\varphi)}(pop_1(push_{n-k+1}(s))) = r :: t$.

Example 10. *In the figure below, we illustrate the definition of $\mathcal{M}$ on the scheme $\mathcal{S}_U$.*



The first line of the definition of $\xrightarrow[\mathcal{M}]{}$ corresponds to the case of an N-stack. To simulate the application of a production rule $A\, x_1 \cdots x_n \xrightarrow{a} e$ on the term encoded by an A-stack s, we simply push the right-hand side e of the production on top of s. The correctness of this rule directly follows from the definition of $[\![\,\cdot\,]\!]$. Doing so, a term starting with a variable may be pushed on top of the stack, *e.g.* when applying the production rule $F\,\varphi\,x\,y \xrightarrow{)} \varphi\,(H\,y)$. Indeed, we need to retrieve the value of the head variable in order to simulate the next transition of $\mathcal{S}$: the second and third lines of the definition are normalisation rules that aim at replacing the variable at the head of the top of the stack (for instance, in the 5th stack of Example 10 the variable φ) by its definition (hence not changing the value of the associated term). By iterative application, we eventually end up with an N-stack encoding the same term and we can apply again the first rule.

Proposition 1. $\mathrm{Tree}^{\perp}(\mathcal{S}) = \mathrm{Tree}^{\perp}(\mathcal{M})$.

Sketch: One easily concludes after establishing the following soundness result about the definition of $\xrightarrow[\mathcal{M}]{}$.

- Let s be an N-stack in WStacks and $a \in \Sigma$.
  For any $t \in \mathrm{Terms}(N)$, if $[\![\, s \,]\!] \xrightarrow{a} t$ then $\exists s' \in$ WStacks, $s \xrightarrow[\mathcal{M}]{a} s'$ and $[\![\, s' \,]\!] = t$.
  If $\exists s' \in$ WStacks, $s \xrightarrow[\mathcal{M}]{a} s'$ then $[\![\, s \,]\!] \xrightarrow{a} [\![\, s' \,]\!]$.
- Let $s \in$ WStacks be a φ-stack for $\varphi \in V$ and let $s' \in$ WStacks be a ψ-stack for $\psi \in V \cup N$.
  If $s \xrightarrow[\mathcal{M}]{e} s'$ then $[\![\, s \,]\!] = [\![\, s' \,]\!]$, $ord(\varphi) \leqslant ord(\psi)$ and $|\, top_{n-ord(\varphi)+1}(s)\,| > |\, top_{n-ord(\varphi)+1}(s')\,|$.
- For all $s \in$ WStacks there exists a unique N-stack $s' \in$ WStacks such that $s \xrightarrow[\mathcal{M}]{e*} s'$.

■

From $\mathcal{M}$ we now define an n-CPDA $\mathcal{A} = \langle\, \Sigma, \Gamma, Q, \delta, q_0 \,\rangle$ generating the same tree as $\mathcal{M}$. The set of states Q is equal to $\{\, q_0, q_1, \ldots, q_{\varrho(\mathcal{S})}, q_* \,\}$ where $\varrho(\mathcal{S})$ denotes the maximal arity appearing in $\mathcal{S}$. Intuitively the initial state q_0 is only used to go from $(q_0, [\ldots[\,\perp\,]_1\ldots]_n)$ to $(q_*, [\ldots[\,\perp Z\,]_1\ldots]_n)$; the state q_* is used to mark N-stacks; for $k \in [1, \varrho(\mathcal{S})]$, the state q_k is used to the compute $\mathrm{Arg}_k(\cdots)$. The transitions are given below.

- $\delta(q_0, \perp, \boldsymbol{e}) = (q_*, push_1^Z)$,
- If t starts with $F \in N$ and $F\, x_1 \cdots x_{\varrho(F)} \xrightarrow{a} e \in \mathcal{R}$:
  - $\delta(q_*, t, a) = (q_*, push_1^e)$ if e starts with a symbol in N,
  - $\delta(q_*, t, a) = (q_{\mathrm{rk}(x)}, id)$ if e is a variable $x : o$ (here id is the identity function),
  - $\delta(q_*, t, a) = (q_{\mathrm{rk}(x)}, push_1^e; push_{n-k+1}; pop_1)$ if e starts with a variable x of order $k > 0$.
- If t is a term of the form $\varphi\, t_1 \cdots t_\ell$ for some $\varphi \in V \cup N$:
  - $\delta(q_k, t, \boldsymbol{e}) = (q_{\mathrm{rk}(t_k)}, pop_1; push_1^{t_k})$ if $k \leqslant \ell$ and $t_k : o$,
  - $\delta(q_k, t, \boldsymbol{e}) = (q_{\mathrm{rk}(t_k)}, pop_1; push_1^{t_k, n-h+1})$ if $k \leqslant \ell$ and t_k has order $h > 0$,
  - $\delta(q_k, t, \boldsymbol{e}) = (q_{k-\ell}, collapse)$ if $k > \ell$.

where, for all $t \in \Gamma$, $q_{\mathrm{rk}(t)}$ designates the state $q_{\mathrm{rk}(x)}$ if t starts with a variable x and q_* otherwise, and $op_1; op_2$ means applying op_1 followed by op_2. An equivalent CPDA using only one operation per transition may be obtained by adding intermediary states.

Theorem 1. *For every labeled recursion scheme $\mathcal{S}$ of order-n, there is an n-CPDA $\mathcal{A}$ that generates the same tree. Moreover, the number of states in $\mathcal{A}$ is linear in the maximal arity appearing in $\mathcal{S}$, and its alphabet is of size linear in*

the one of $\mathcal{S}^3$.

Remark 3. *In [18], the authors use Krivine machines [14] as an abstract model to represent the sequence of rewriting of a scheme[4]. A Krivine machine computes the weak head normal form of a λY-term, using explicit substitutions (called here environments). Environments are functions assigning closures to variables, and closures themselves are pairs consisting of a term and an environment. This mutually recursive definition is schematically represented by the grammar $C := (t, \rho)$ and $\rho := \varnothing \mid \rho[x \to C]$ where t is an term of the λY-calculus with free-variable and $\varnothing$ designates the empty environment. The λY-term t_C represented by a closure $C = (t, \rho)$ is inductively defined as t in which every occurrence of a free variable x is replaced by the term $t_{\rho(x)}$.*

A pair $s :: t$ (cf. Notation 2) can be seen as a closure[5] (t, ρ) where $\rho(x)$ is defined for all variables x occurring in t by $\rho(x) = Arg_{\mathrm{rk}(x)}(s)$. With this view in mind and up to the translation of schemes into equivalent λY-terms, the LTS *$\mathcal{M}$ faithfully simulates the Krivine machine presented in [18]. Note that the correspondence is facilitated by the use of labeled schemes.*

This remark also allows us to inherit the simplifications of [18] for the decidability of CPDA parity games.

IV. Safe Higher-order Recursion Schemes

In this section, we consider a syntactic subfamily of recursion schemes called the *safe recursion schemes*. The *safety* constraint was introduced in [10] but was already implicit in the work of Damm [6] (see also [7, p. 44] for a detailed presentation). This restriction constrains the way variables are used to form argument subterms of the rules' right-hand sides.

Definition 5 ([10])**.** *A recursion scheme is* safe *if no right-hand side contains an argument-subterm of order k containing a variable of order strictly less than k.*

For instance, the scheme in Example 3 is safe. On the other hand, the scheme $\mathcal{S}_U$ of Example 4 is not because the production $F\,\varphi\,x\,y \xrightarrow{(} F(F\varphi x)y(Hy)$ contains in its right-hand side the argument subterm $F\varphi x : \mathrm{o} \to \mathrm{o}$ of order-1 which contains the variable $x : \mathrm{o}$ of order-0. Urzyczyn conjectured that (a slight variation of) the tree T_U generated by $\mathcal{S}_U$, though generated by a order-2 scheme, could not be generated by any *safe* scheme. This conjecture was recently proved by Parys [16].

Remark 4. *In [10], [11], the notion of safety is only defined for* homogeneous *schemes. A type is said to be* homogeneous *if it is either ground or equal to $\tau_1 \to \cdots \to \tau_n \to \mathrm{o}$ where the τ_i's are homogeneous and $\mathrm{ord}(\tau_1) \geqslant \cdots \geqslant \mathrm{ord}(\tau_n)$. By extension, a scheme is homogeneous if all its non-terminal symbols have homogeneous types. For instance $(\mathrm{o} \to \mathrm{o}) \to \mathrm{o} \to \mathrm{o}$ is an homogeneous type whereas $\mathrm{o} \to (\mathrm{o} \to \mathrm{o}) \to \mathrm{o}$ is not. We will see in Proposition 2 that dropping the homogeneity constraint in the definition of safety does not change the family of generated trees.*

A. Safety and the Translation from Schemes to CPDA

In [10], [11], the motivation for considering the safety constraint was that safe schemes can be translated into a subfamily of the collapsible automata, namely higher-order pushdown automata. An order-k pushdown automaton is an order-k CPDA that does not use the collapse operation (hence, links are useless).

Theorem 2 below shows that the translation of recursion schemes into collapsible automata presented in Section III, when applied to a safe scheme, yields an automaton in which links are not really needed. Obviously the automaton performs the collapse operations but whenever it is applied to an order-k link its target is the $(k-1)$-stack below the top $(k-1)$-stack. Hence any collapse operation can safely be replaced by a pop_k operation. In doing so, we re-obtain the translation of safe (homogeneous) schemes into higher-order pushdown automata presented in [11].

Definition 6. *A CPDA is* link-free *if for every configuration (p, s) reachable from the initial configuration and for every transition $\delta(p, top_1(s), a) = (q, collapse)$, we have $collapse(s) = pop_\ell(s)$ where ℓ is the order of the link attached to $top_1(s)$.*

Theorem 2. *The translation of Section III applied to a safe recursion scheme yields a link-free collapsible automaton.*

Sketch: We present the ingredients of the proof only at order-2. For the general case, the ideas are similar but lead to more technicalities.

Let us first introduce some notations. Let $(q, s = [s_1 \dots s_m]_2)$ be a configuration of $\mathcal{A}$ reachable from the initial configuration. For $i \in [1, m]$ and $j \in [1, |s_i|]$, we denote by $r(i, j)$, $t(i, j)$ and $o(i, j)$ respectively the j-th symbol of stack s_i, the target (if defined) in $[1, i-1]$ of its link and the order (if defined) of this link. By definition of $\mathcal{A}$, $t(i, j)$ and $o(i, j)$ are defined iff $r(i, j)$ is a term of order $k > 0$ and in this case $o(i, j)$ is equal to $2 - k + 1$

Moreover for $i \in [2, m]$, we let ℓ_i be the smallest index at which s_{i-1} and s_i have a different symbol (or $|s_i| + 1$ if no such index exists).

The stack s satisfies the following properties:

1) for all $i \in [1, |s_1|]$, $t(1, i)$ is undefined;
2) for all $i \in [2, m]$, $\ell_i \leqslant |s_{i-1}|$ and for all $i \in [2, m-1]$, $\ell_i \leqslant |s_i|$;
3) for all $i \in [2, m]$ and $1 \leqslant j < \ell_i$, $t(i, j) = t(i-1, j)$;

[3]The size of a scheme is defined as the sum of the sizes of the left and right hand sides of the rewriting rules. In particular it is larger than the sum of the sizes of all argument subterms of right hand sides of the rules.

[4]The authors work with the equivalent formalism of the λY-calculus.

[5]to represent applicative terms over N instead of λY-terms.

4) for all $i \in [2,m]$ with $\ell_i \leqslant |s_i|$, $r(i,\ell_i)$ does not contain a variable of order 0 and is an argument subterm of $r(i-1,\ell_i)$ and if $r(i,\ell_i)$ is of order 1 then $t(i,\ell_i) = i-1$;
5) for all $i \in [2,m]$ with $j \in [\ell_i+1, |s_i|]$, $t(i,j)$ is undefined;
6) if $m \geqslant 2$ then $\ell_m = |s_m|+1$ iff $top_1(s) = \varphi t_1 \dots t_h$, $q = q_k$ for some $k \in [1,h]$ such that $ord(t_k) = 1$.

These properties are proved by induction on the length of the shortest path in the LTS from the initial configuration to (q,s) and by inspection of the transitions of $\mathcal{A}$.

Inspecting the transitions of $\mathcal{A}$, a *collapse* operation can only be performed if $q = q_k$ and $top_1(s) = \varphi\, t_1 \dots t_h$ with $k > h$ and $\varphi : (\tau_1, \dots, \tau_m, o)$. Thanks to Definition 1, $\varphi\, t_1 \dots t_h$ is of order-1. Property 5 implies that ℓ_m is either equal to $|s_m|$ or to $|s_m|+1$. Property 6 implies $\ell_m \neq |s_m|+1$ as otherwise we would have $k \leqslant h$. Thus, we have $\ell_m = |s_m|$ and by Property 4, $collapse(s) = pop_2(s)$. ∎

We get the following corollary extending (by dropping the homogeneity assumption) a previous result from [11].

Corollary 1. *Order-k safe schemes and order-k pushdown automata generate the same trees.*

B. Damm's View of Safety

The safety constraint may seem unnatural and purely *ad-hoc*. Inspired by the constraint of derived types of Damm, we introduce a more natural constraint, *Damm-safety*, which leads the same family of trees [6].

Damm-safety syntactically restricts the use of partial application: in any argument subterm of a right-hand side if one argument of some order-k is provided then all arguments of order-k must also be provided. For instance if $\varphi : (o \to o) \to (o \to o) \to o \to o \to o$, $f : o \to o$ and $c : o$, the terms φ, $\varphi\, f\, f$ and $\varphi\, f\, f\, c\, c$ can appear as argument subterms in a Damm-safe scheme but $\varphi\, f$ and $\varphi\, f\, f\, c$ are forbidden.

Definition 7 ([6])**.** *A recursion scheme is* Damm-safe *if it is homogeneous and all argument-subterms appearing in a right hand-side are of the form $\varphi\, t_1 \cdots t_k$ with $\varphi : \tau_1 \to \cdots \to \tau_n \to o$ and either $k = 0$, $k = n$ or $ord(\tau_k) > ord(\tau_{k+1})$.*

As in Damm-safe scheme all argument subterms of an argument subterm of order-k appearing in a right-hand side have at least order-k, it is easy to see that Damm-safety implies the safety constraint. However, the safety constraint, even when restricted to homogeneous schemes, is less restrictive than Damm-safety. Consider for instance a variable $x : o$ and non-terminals $G : o \to o \to o$ and $C : o$, then $G\, x$ cannot appear as an argument-subterm in a safe scheme but $G\, C$ can. As $G\, C$ does not satisfy Damm-safety constraint, safety is syntactically more permissive than Damm-safety. However unsurprisingly, any safe scheme can be transformed into an equivalent Damm-safe scheme of the same order. The transformation consists in converting the safe scheme into a higher-order pushdown automaton (Corollary 1) and then converting this automaton back to a scheme using the translation of [11]. In fact, this translation of higher-order pushdown automata into safe schemes produces Damm-safe schemes.

Proposition 2. *Damm-safe schemes are safe and for every safe scheme, there exists a Damm-safe scheme of the same order generating the same tree.*

V. Effective Selection

Let $\varphi(X_1, \cdots, X_\ell)$ be a monadic second order (MSO) formula with ℓ second-order free variables, and let t be a term over a ranked alphabet Σ. The *MSO selection problem* is to decide whether the formula $\exists X_1 \dots \exists X_\ell\ \varphi(X_1, \cdots, X_\ell)$ holds in t, and in this case to give a term t_φ over the ranked alphabet $\Xi = \Sigma \times \{0,1\}^\ell$ (we take $\varrho(a,(b_1,\dots,b_\ell)) = \varrho(a)$) such that the following holds:

1) $t = \pi(t_\varphi)$ where π is the alphabetical morphism from $\overrightarrow{\Xi}$ to $\overrightarrow{\Sigma}$ defined by $\pi((a,\overline{b})) = a$ for $a \in \Sigma$ with $\varrho(a) = 0$ and $\pi((a,\overline{b})_i) = a_i$ for $a \in \Sigma$ with $\varrho(a) > 0$ and $i \in [1, \varrho(a)]$. Intuitively, t_φ is obtained by marking every node in t by a vector of ℓ booleans. Indeed for all non-leaf node u, there exists a unique element $(c,\overline{b}) \in \Xi$ such that for all $x \in \overrightarrow{(c,\overline{b})}$, ux is in t_φ. The tuple $\overline{b} \in \{0,1\}^\ell$ is the label of the node u of t. The label of a non-leaf node u of t is denoted b_u
2) The formula $\varphi(X_1 \leftarrow U_1, \dots, X_\ell \leftarrow U_\ell)$ holds in t where $\forall 1 \leqslant i \leqslant \ell$, $U_i = \{u \in t \mid b_u(i) = 1\}$.

Intuitively, the second point states that this marking exhibits a valuation of the X_i for which φ holds in t. We refer to t_φ as a *selector for φ in t*.

Let $\mathcal{R}$ be a class of generators of terms. We say that $\mathcal{R}$ has the *effective MSO selection property* if there is an algorithm that transforms any pair $(R, \varphi(X_1, \dots, X_\ell))$ with $R \in \mathcal{R}$ into some $R_\varphi \in \mathcal{R}$ (if exists) such that the term generated by R_φ is a selector for φ in the term generated by R.

Theorem 3. *Labeled recursion schemes as well as CPDA have the effective MSO selection property.*

The proof of Theorem 3 is highly non-trivial and requires a precise analysis of winning strategies in parity games played over terms generated by CPDA (the key argument is that winning strategies can be embedded into the CPDA generating the term). We do not believe that a proof of the statement for labeled recursion schemes can be obtained without using an automaton model, and we think that it shows the usefulness of CPDA in the study of logical properties of schemes.

Remark 5. *A similar statement for* safe *schemes can be deduced from [8], [3], [5]. However the machinery for general schemes is much more involved.*

In [2] a much weaker notion, *MSO-reflectivity*, was considered. A class of generators of terms is MSO-reflective if it has the effective MSO selection property for those formula $\varphi(X)$ of the form $\varphi(X) \equiv x \in X \Leftrightarrow \psi(x)$ where $\psi(x)$ is an MSO formula with a single first-order free variable (note that in this case, there is a unique valuation of X that makes $\varphi(X)$ holds). The main result of [2] follows from Theorem 3.

Corollary 2. *Labeled recursion schemes as well as CPDA have the effective MSO-reflectivity property.*

Remark 6. *A variant of selection [17] ask for existence of a formula $\psi(X_1, \dots X_\ell)$ that is a* selector *for $\varphi(X_1, \dots X_\ell)$ in t in the following sense. Either neither of the formulas $\exists X_1 \dots \exists X_\ell \ \varphi(X_1, \cdots, X_\ell)$ and $\exists X_1 \dots \exists X_\ell \ \psi(X_1, \cdots, X_\ell)$ holds in t or ψ defines a unique tuple $(U_1, \cdots, U_\ell)$ and this tuple also satisfies φ. In [4] it is shown that a selector does not always exist in general, and the counter-example is for a tree generated by a (safe) recursion scheme.*

A degenerated version of selection is model-checking. Theorem 1 together with a careful analysis of the complexity of parity games on CPDA lead the same complexity as in [13].

Corollary 3. *The μ-calculus model-checking of trees generated by recursion schemes is* polynomial *under the assumption that the arity of types and the formula are bounded above by a constant.*

Acknowledgements: This work was supported by the following projects: AMIS (ANR 2010 JCJC 0203 01 AMIS) and FREC (ANR 2010 BLAN 0202 02 FREC).

REFERENCES

[1] C. Broadbent. *On Collapsible Pushdown Automata, their Graphs and the Power of Links*. PhD thesis, University of Oxford, Forthcoming.

[2] C. Broadbent, A. Carayol, C.-H. L. Ong, and O. Serre. Recursion schemes and logical reflexion. In *Proc. of LICS'10*, pages 120–129. IEEE, 2010.

[3] A. Carayol. *Automates infinis, logiques et langages*. PhD thesis, Université de Rennes 1, 2006.

[4] A. Carayol, C. Löding, D. Niwiński, and I. Walukiewicz. Choice functions and well-orderings over the infinite binary tree. *Central European Journal of Mathematics*, 8(4):662–682, 2010.

[5] A. Carayol and M. Slaats. Positional strategies for higher-order pushdown parity games. In *Proc. of MFCS'08*, volume 5162 of *LNCS*, pages 217–228. Springer, 2008.

[6] W. Damm. The IO- and OI-hierarchies. *Theoret. Comput. Sci.*, 20:95–207, 1982.

[7] J. de Miranda. *Structures generated by higher-order grammars and the safety constraint*. PhD thesis, University of Oxford, 2006.

[8] S. Fratani. *Automates à piles de piles ... de piles*. PhD thesis, Université de Bordeaux, 2006.

[9] M. Hague, A. S. Murawski, C.-H. L. Ong, and O. Serre. Collapsible pushdown automata and recursion schemes. In *Proc. of LICS'08*, pages 452–461. IEEE, 2008.

[10] T. Knapik, D. Niwiński, and P. Urzyczyn. Deciding monadic theories of hyperalgebraic trees. In *Proc. of TLCA'01*, volume 2044 of *LNCS*, pages 253–267. Springer, 2001.

[11] T. Knapik, D. Niwiński, and P. Urzyczyn. Higher-Order Pushdown Trees Are Easy. In *Proc. of FoSSaCS'02*, volume 2303 of *LNCS*, pages 205–222. Springer, 2002.

[12] T. Knapik, D. Niwiński, P. Urzyczyn, and I. Walukiewicz. Unsafe grammars and panic automata. In *Proc. of ICALP'05*, volume 3580 of *LNCS*, pages 1450–1461. Springer, 2005.

[13] N. Kobayashi and C.-H. L. Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proc. of LICS'09*, pages 179–188. IEEE, 2009.

[14] J.-L. Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3):199–207, 2007.

[15] C.-H. L. Ong. On model-checking trees generated by higher-order recursion schemes. In *Proc. of LICS'06*, pages 81–90. IEEE, 2006.

[16] P. Parys. On the Significance of the Collapse Operation. In *Proc. of LiCS'12*. IEEE, 2012.

[17] A. Rabinovich and A. Shomrat. Selection and uniformization problems in the monadic theory of ordinals: A survey. In *Pillars of Computer Science*, volume 4800 of *LNCS*, pages 571–588. Springer, 2008.

[18] S. Salvati and I. Walukiewicz. Krivine machines and higher-order schemes. In *Proc. of ICALP'11*, volume 6756 of *LNCS*, pages 162–173. Springer, 2011.

APPENDIX

A. Proofs Omitted in Section II

1) Labeled Recursion Schemes vs *(Classical) Recursion Schemes:*

We recall the notion of recursion schemes as it is usually considered in the literature (see *e.g.* [9]).

For each type τ, we assume an infinite set V_τ of variables of type τ, such that V_{τ_1} and V_{τ_2} are disjoint whenever $\tau_1 \neq \tau_2$, and we write V for the union of those sets V_τ as τ ranges over types. We use letters $x, y, \varphi, \psi, \chi, \xi, \ldots$ to range over variables.

A (deterministic) *recursion scheme* is a 5-tuple $\mathcal{S} = \langle A, N, \mathcal{R}, Z, \bot \rangle$ where

- A is a ranked alphabet of *terminals* and $\bot$ is a distinguished terminal symbol of arity 0 (and hence of ground type) that does not appear in any production rule,
- N is a finite set of typed *non-terminals*; we use upper-case letters $F, G, H, \ldots$ to range over non-terminals,
- $Z \in N$ is a distinguished *initial symbol* of type o which does not appear in any right-hand side of a production rule,
- $\mathcal{R}$ is a finite set of *production rules*, one for each non-terminal $F : (\tau_1, \cdots, \tau_n, o)$, of the form

$$F\, x_1 \cdots x_n \;\rightarrow\; e$$

where the x_i are distinct variables with $x_i : \tau_i$ for $i \in [1, n]$ and e is a ground term in $\mathrm{Terms}((A \backslash \{\bot\}) \cup (N \backslash \{Z\}) \cup \{ x_1, \ldots, x_n \})$. Note that the expressions on either side of the arrow are terms of *ground* type.

As for labelled schemes, the *order* of a recursion scheme is defined to be the highest order of (the types of) its non-terminals.

A recursion scheme $\mathcal{S}$ induces a rewriting relation, denoted $\rightarrow_{\mathcal{S}}$, over $\mathrm{Terms}(A \cup N)$. Informally, $\rightarrow_{\mathcal{S}}$ replaces any ground subterm $F\, t_1 \ldots t_{\varrho(F)}$ starting with a non-terminal F by the right-hand side of the production rule $F\, x_1 \cdots x_n \;\rightarrow\; e$ in which the occurrences of the "formal parameter" x_i are replaced by the actual parameter t_i for $i \in [1, \varrho(F)]$.

The term $M[t/x]$ obtained by replacing a variable $x : \tau$ by a term $t : \tau$ over $A \cup N$ in a term M over $A \cup N \cup V$ is defined[6] by induction on M by taking $\varphi[t/x] = \varphi$ for $\varphi \neq x \in A \cup N \cup V$, $x[t/x] = t$ and $(t_1\, t_2)[t/x] = t_1[t/x]\, t_2[t/x]$.

The rewriting system $\rightarrow_{\mathcal{S}}$ is defined by induction using the following rules:

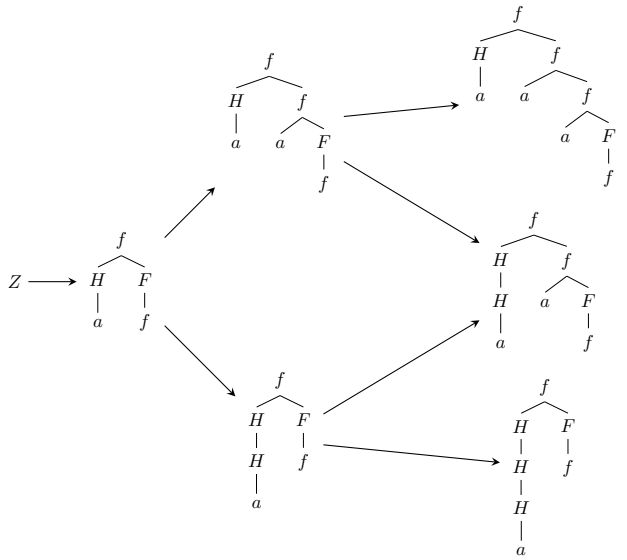
- (*Substitution*) $F t_1 \cdots t_n \rightarrow_{\mathcal{S}} e[t_1/x_1, \cdots, t_n/x_n]$ where $F x_1 \cdots x_n \rightarrow e$ is a production rule of $\mathcal{S}$.
- (*Context*) If $t \rightarrow_{\mathcal{S}} t'$ then $(st) \rightarrow_{\mathcal{S}} (st')$ and $(ts) \rightarrow_{\mathcal{S}} (t's)$.

Example 11. Let $\mathcal{S}$ be the order-2 recursion scheme with non-terminals $\{Z : o,\, H : (o, o),\, F : ((o, o, o), o)\}$, variables $\{z : o,\, \varphi : (o, o, o)\}$, terminals $A = \{f, a\}$ of arity 2 and 0 respectively, and the following rewrite rules:

$$\begin{array}{rcl} Z & \rightarrow & f(H\, a)(F\, f) \\ H\, z & \rightarrow & H\, (H\, z) \\ F\, \varphi & \rightarrow & \varphi\, a\, (F\, \varphi) \end{array}$$

The figure below depicts the first rewriting steps of $\rightarrow_S$ starting from the initial symbol Z.

[6]Note that t does not contain any variables and hence we do not need to worry about capture of variables.



As illustrated above, the relation $\rightarrow_{\mathcal{S}}$ is confluent, *i.e.* for all ground terms t,t_1 and t_2, if $t \rightarrow_{\mathcal{S}}^* t_1$ and $t \rightarrow_{\mathcal{S}}^* t_2$ (here $\rightarrow_{\mathcal{S}}^*$ denotes the transitive closure of $\rightarrow_{\mathcal{S}}$), then there exists t' such that $t_1 \rightarrow_{\mathcal{S}}^* t'$ and $t_2 \rightarrow_{\mathcal{S}}^* t'$. The proof of this statement is similar to proof of the confluence of the lambda-calculus.

Informally the *value tree* of (or the tree *generated* by) a recursion scheme $\mathcal{S}$, denoted $[\![\, \mathcal{S} \,]\!]$, is a (possibly infinite) term, *constructed from the terminals in* A, that is obtained as the "limit" of the set of all terms that can obtained by iterative rewriting from the initial symbol Z.

The terminal symbol $\bot : o$ is used to formally restrict terms over $A \cup N$ to their terminal symbols. We define a map $(\cdot)^\bot : \mathrm{Terms}(A \cup N) \longrightarrow \mathrm{Terms}(A)$ that takes an applicative term and replaces each non-terminal, together with its arguments, by $\bot : o$. We define $(\cdot)^\bot$ inductively as follows, where a ranges over A-symbols, and F over non-terminals in N:

$$\begin{array}{rcl} a^\bot & = & a \\ F^\bot & = & \bot \\ (st)^\bot & = & \begin{cases} \bot & \text{if } s^\bot = \bot \\ (s^\bot t^\bot) & \text{otherwise.} \end{cases} \end{array}$$

Clearly if $t \in \mathrm{Terms}(A \cup N)$ is of ground type then $t^\bot \in \mathrm{Terms}(A)$ is of ground type as well.

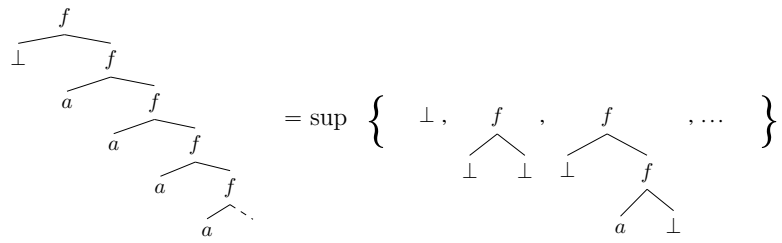
Terms built over A can be partially ordered by the approximation ordering $\preccurlyeq$ defined for all terms t and t' over A by $t \preccurlyeq t'$ if $t \cap (\overrightarrow{A} \backslash \{\bot\})^* \subseteq t'$. In other terms, t' is obtained from t by substituting some occurrences of $\bot$ by arbitrary terms over A.

The set of terms over A together with $\preccurlyeq$ form a *directed complete partial order*. Meaning that any directed subset D of $\mathrm{Terms}(A)$ (*i.e.* D is not empty and for all $x, y \in D$, there exists $z \in D$ such that $x \preccurlyeq z$ and $y \preccurlyeq z$) admits a *supremum*, denoted $\sup D$.

Clearly if $s \rightarrow_{\mathcal{S}} t$ then $s^\bot \preccurlyeq t^\bot$. The confluence of $\rightarrow_{\mathcal{S}}$ implies that the set $\{\, t^\bot \mid Z \rightarrow_{\mathcal{S}}^* t \,\}$ is directed. Hence the *value tree* of (or the tree *generated* by) $\mathcal{S}$ can be defined as its supremum.

$$[\![\, \mathcal{S} \,]\!] \;=\; \sup\{\, t^\bot \mid Z \rightarrow_{\mathcal{S}}^* t \,\}.$$

Example 12. The value tree of the recursion scheme $\mathcal{S}$ of Example 11 is:



The following theorem relates both notions of schemes.

Theorem 4. *The recursion schemes and the labeled recursion schemes generate the same terms. Moreover the translations are linear and preserves order and arity.*

Proof: Let $\mathcal{S} = \langle A, N, \mathcal{R}, Z, \bot \rangle$ be a recursion scheme. We define a labeled recursion scheme $\mathcal{S}' = \langle \overrightarrow{A}, N', \mathcal{R}', Z, \bot \rangle$ generating the term $[\![\mathcal{S}]\!]$. For each terminal symbol $f \in A$, we introduce a non-terminal symbol, denoted $\overline{f}$: $\underbrace{o \to \cdots \to o}_{\varrho(f)} \to o$. The set N' of non-terminal symbols of $\mathcal{S}'$ is $N \cup \{\overline{f} \mid f \in A\} \cup \{X\}$ where X is assumed to be a fresh non-terminal. With a term t over $A \cup N$, we associate the term $\overline{t}$ over N' obtained by replacing every occurrence of a terminal symbol f by its nonterminal counterpart $\overline{f}$. The production rules $\mathcal{R}'$ of $\mathcal{S}'$ are:

$$\begin{array}{ll} & \{F\,x_1 \cdots x_n \xrightarrow{e} \overline{e} \mid F\,x_1 \cdots x_n \longrightarrow e \in \mathcal{R}\} \\ \cup & \{\overline{f}\,x_1 \cdots x_{\varrho(f)} \xrightarrow{f_i} x_i \mid f \in A \text{ with } \varrho(f) > 0 \text{ and } i \in [1, \varrho(f)]\} \\ \cup & \{\overline{c} \xrightarrow{c} X \mid c \in A \text{ with } \varrho(c) = 0\} \end{array}$$

Conversely, let A be ranked alphabet and let $\mathcal{S} = \langle \overrightarrow{A}, N, \mathcal{R}, Z, \bot \rangle$ be a labeled recursion scheme generating a ranked tree. We define a recursion scheme $\mathcal{S}' = \langle A, N, \mathcal{R}', Z, \bot \rangle$ generating the same term as $\mathcal{S}$. The set of production rules of $\mathcal{S}'$ are defined as follows:

- If $F\,x_1 \cdots x_n \xrightarrow{e} e$ belongs to $\mathcal{R}$ (in this case it is the only rule starting with F) then $F\,x_1 \cdots x_n \to e$ belongs to $\mathcal{R}'$.
- If, for some c of arity 0, $F\,x_1 \cdots x_n \xrightarrow{c} e$ belongs to $\mathcal{R}$ (in this case it is the only rule starting with F and e starts with a non-terminal that has no rule in $\mathcal{R}$) then $F\,x_1 \cdots x_n \to c$ belongs to $\mathcal{R}'$.
- If, for some $f \in A$ of arity $\varrho(f) > 0$, $F\,x_1 \cdots x_n \xrightarrow{f_i} e_i$ belongs to $\mathcal{R}$ for all $1 \leqslant i \leqslant \varrho(f)$, then $F\,x_1 \cdots x_n \to f\,e_1 \cdots e_{\varrho(f)}$ belongs to $\mathcal{R}'$.

■

2) Extra Examples of Labeled Recursion Schemes:

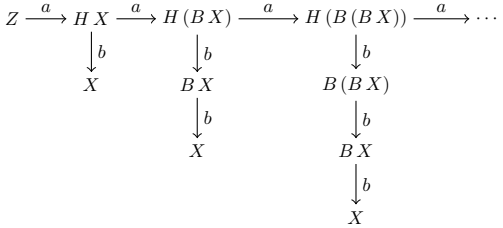
Due to space limitation we could only give two examples of labelled recursion schemes in the main body of the paper. We present here some extra example to illustrate the mechanism of labelled recursion schemes as well as their expressive power.

Example 13. *Let T_0 be the tree corresponding to the deterministic context-free language $\mathrm{Pref}(\{a^n b^n \mid n \geqslant 0\})$. As it is the case for all prefix-closed deterministic context-free languages (see [A2], [A3] or Theorem 1 at order 1) , T_0 is generated by an order-1 scheme $\mathcal{S}_0$.*

$$\begin{array}{llll} Z & \xrightarrow{a} & H\,X & \qquad H\,x \xrightarrow{a} H\,(B\,z) \\ B\,x & \xrightarrow{b} & x & \qquad H\,x \xrightarrow{b} x \end{array}$$

with $Z, X : o$ and $H, B : o \to o$.

The tree generated by $\mathcal{S}_0$ is given below:



Example 14. *Following the same ideas as for $\mathcal{S}_1$ (see Example 3), the order-2 scheme $\mathcal{S}_{\exp}$ given below defines the tree $T_{\exp} = \mathrm{Pref}(\{a^n c b^{2^n} \mid n \geqslant 0\})$.*

$$\begin{array}{llll} Z & \xrightarrow{e} & F\,B & \qquad F\,\varphi \xrightarrow{a} F\,(D\,\varphi) \qquad D\,\varphi\,x \xrightarrow{e} \varphi\,(\varphi\,x) \\ B\,x & \xrightarrow{b} & x & \qquad F\,\varphi \xrightarrow{c} \varphi\,X \end{array}$$

with $Z, X : o$, $B : o \to o$, $D : (o \to o, o, o)$ and $F : (o \to o, o)$. If we denote by $\underline{D^n\,B}$ the term of type $o \to o$ defined by $\underline{D^0 B} = B$ and $\underline{D^{n+1}\,B} = D\,(\underline{D^n\,B})$ for $n \geqslant 0$, we have $Z \overset{a^n}{\Longrightarrow} F\,\underline{D^n\,B}$. As D intuively doubles its argument, $\underline{D^n\,B}$ behaves as B^{2^n} for $n \geqslant 0$. In particular, $\underline{D^n\,B}\,X$ reduces by b^{2^n} to X.
For all $n \geqslant 0$, we have:

$$Z \overset{a^n}{\Longrightarrow} F\,\underline{D^n\,B} \xrightarrow{c} \underline{D^n\,B}\,X \overset{b^{2^n}}{\Longrightarrow} X.$$

Example 15. *At order $k+1 \geqslant 1$, we can define the tree $T_{\exp_k} = \mathrm{Pref}(\{a^n\,c\,b^{\exp_k(n)} \mid n \geqslant 0\})$ where we let $\exp_0(n) = n$ and $\exp_{k+1}(n) = 2^{exp_k(n)}$ for $k \geqslant 0$. We illustrate the idea by giving an order-3 scheme generating $T_{\exp_2} = \mathrm{Pref}(\{a^n\,c\,b^{2^{2^n}} \mid n \geqslant 0\})$.*

$$\begin{array}{llll} Z & \xrightarrow{e} & F\,D_1 & \qquad F\,\psi \xrightarrow{a} F\,(D_2\,\psi) \\ D_2\,\psi\,\varphi\,x & \xrightarrow{e} & (\psi(\psi\,\varphi))x & \qquad B\,x \xrightarrow{b} x \\ F\,\varphi & \xrightarrow{c} & \varphi\,B\,X & \qquad D_1\,\psi\,x \xrightarrow{e} \psi\,(\psi\,x) \end{array}$$

with $Z, X : o$, $B : o \to o$, $F : ((o \to o, o, o), o)$, $D_1 : (o \to o, o, o)$ and $D_2 : ((o \to o, o, o), o \to o, o, o)$. If we denote by $\underline{D_2^n\,D_1}$ the term of type $(o \to o, o, o)$ defined by $\underline{D_2^0\,D_1} = D_1$ and $\underline{D_2^{n+1}\,D_1} = D_2\,\underline{D_2^n\,D_1}$ for $n \geqslant 0$, we have $Z \overset{a^n}{\Longrightarrow} F\,\underline{D_2^n\,D_1}$. As D_2 intuitively double its argument with each application, $\underline{D_2^n\,D_1}$ behaves as $D_1^{2^n}$ and hence $D_1^{2^n}\,B$ behaves as $B^{2^{2^n}}$.
We have $\forall n \geqslant 0$: $Z \overset{a^n}{\Longrightarrow} F\,\underline{D_2^n\,D_1} \xrightarrow{c} \underline{D_2^n\,D_1}\,B\,X \overset{b^{2^{2^n}}}{\Longrightarrow} X$.

B. Proofs Omitted in Section III

1) Justification of Definition 4:

We start with a fact that justifies the second part of Definition-4

Definition 4. *The term associated with a well-formed φ-stack $s \in \mathrm{Stacks}$ with $\varphi \in N \cup V$ is*

$$[\![\, s \,]\!] \stackrel{\text{def}}{=} [\![\, pop_1(s) :: \varphi \,]\!][\![\, Arg_1(s) \,]\!] \cdots [\![\, Arg_{\varrho(\varphi)}(s) \,]\!].$$

Equiv., if $top_1(s) : o$ then: $[\![\, s \,]\!] = [\![\, pop_1(s) :: top_1(s) \,]\!]$.
If $top_1(s) : \tau_1 \to \ldots \to \tau_\ell \to \mathrm{o}$ then:
$[\![\, s \,]\!] = [\![\, pop_1(s) :: top_1(s) \,]\!] [\![\, Arg_1(collapse(s)) \,]\!] \cdots$
$[\![\, Arg_\ell(collapse(s)) \,]\!]$.

Fact 1. *Let s be a well-formed φ-stack. If $top_1(s) : o$ then:*

$$[\![\, s \,]\!] = [\![\, pop_1(s) :: top_1(s) \,]\!].$$

If $top_1(s) : \tau_1 \to \ldots \to \tau_\ell \to \mathrm{o}$ then:

$$[\![\, s \,]\!] = [\![\, pop_1(s) :: top_1(s) \,]\!] [\![\, Arg_1(collapse(s)) \,]\!] \cdots [\![\, Arg_\ell(collapse(s)) \,]\!].$$

Proof: The first case ($top_1(s) : o$) is immediate. Assume that $top_1(s)$ is equal to $\varphi\, t_1 \cdots t_n$ with $\varphi \in N \cup V$ of type $\tau_1 \to \ldots \to \tau_{\varrho(\varphi)} \to \mathrm{o}$ and for all $i \in [1, n]$, $t_i \in \Gamma$ of type τ_i. Note that $\ell = \varrho(\varphi) - n$. We have:

$$\begin{aligned} [\![\, s \,]\!] &\stackrel{\text{def}}{=} \underbrace{[\![\, pop_1(s) :: \varphi \,]\!][\![\, \mathrm{Arg}_1(s) \,]\!] \cdots [\![\, \mathrm{Arg}_n(s) \,]\!]}_{[\![\, pop_1(s) :: \varphi\, t_1 \cdots t_n \,]\!]} [\![\, \mathrm{Arg}_{n+1}(s) \,]\!] \cdots [\![\, \mathrm{Arg}_{\varrho(\varphi)}(s) \,]\!] \\ &= [\![\, pop_1(s) :: top_1(s) \,]\!] [\![\, \mathrm{Arg}_1(collapse(s)) \,]\!] \cdots [\![\, \mathrm{Arg}_{\varrho(\varphi)-n=\ell}(collapse(s)) \,]\!] \end{aligned}$$

■

2) Proof of Lemma 1:

Lemma 1. *We have the following properties:*

1) *For all φ-stacks $s \in \mathrm{WStacks}$ with $\varphi \in V \cup N$ of type $\tau_1 \to \ldots \to \tau_{\varrho(\varphi)} \to \mathrm{o}$ and for all $k \in [1, \varrho(\varphi)]$, $Arg_k(s)$ is equal to some $r :: t \in \mathrm{CStacks}$ with t of type τ_k.*
2) *For all $s :: t \in \mathrm{CStacks}$ with $t : \tau \in \Gamma$, $[\![\, s :: t \,]\!]$ is a term in $\mathrm{Terms}_\tau(N)$.*
3) *For all $s \in \mathrm{WStacks}$, $[\![\, s \,]\!]$ belongs to $\mathrm{Terms}(N)$.*

Proof: We start proving the first point and then use it to obtain the second one. Combining them, we finally prove the last point.

(1) We proceed by induction on the size of $s \in \mathrm{WStacks}$. The base case considers the stack $[\cdots[\bot Z]_1 \cdots]_n$. As $\varrho(Z) = 0$, there is nothing to prove.

Fix some stack s and assume that the property holds for all stacks smaller than $s \in \mathrm{WStacks}$. Let $\varphi\, t_1 \cdots t_\ell : \tau$ be the top symbol of s with $\varphi \in N \cup V$, $\ell \in [1, \varrho(\varphi)]$ and $t_i \in \Gamma$ for all $i \in [1, \ell]$. If φ is of type $\tau_1 \to \ldots \to \tau_{\varrho(\varphi)} \to \mathrm{o}$ then for all $i \in [1, \ell]$, t_i is of type τ_i and τ is the type $\tau_{\ell+1} \to \ldots \to \tau_{\varrho(\varphi)} \to \mathrm{o}$.

If $k \leqslant \ell$, $\mathrm{Arg}_k(s) \stackrel{\text{def}}{=} pop_1(s) :: t_k$ and there is nothing to prove. If $\varrho(\varphi) \geqslant k > \ell$, $\mathrm{Arg}_k(s) \stackrel{\text{def}}{=} \mathrm{Arg}_{k-\ell}(collapse(s))$. To conclude by induction, the only thing we have to prove that $\mathrm{Arg}_{k-\ell}(collapse(s))$ is well defined. As $ord(\tau) > 0$, we have by definition of WStacks that $collapse(s)$ is well-defined and that its top symbol starts with a symbol ψ of type τ. As $|collapse(s)| < |s|$ and as $\varrho(\psi) = \varrho(\varphi) - \ell \geqslant k - \ell \geqslant 1$, we have by induction hypothesis that $\mathrm{Arg}_{k-\ell}(collapse(s))$ is well-defined and is equal to some $r :: t \in \mathrm{CStacks}$ with $t \in \Gamma$ of type $\tau_{k-\ell+\ell} = \tau_k$.

(2) We proceed by induction on the size of $s :: t$. The base case deals with $[\cdots[\,\bot\,]_1 \cdots]_n :: Z$. As $[\![\, [\,]_n :: Z \,]\!] \stackrel{\text{def}}{=} Z$, the property holds.

Assume that the property holds for all elements of CStacks smaller than some $s :: t \in \mathrm{CStacks}$ with $t : \tau$. Let us show that $[\![\, s :: t \,]\!]$ is of type τ. The case where $t \in N$ is trivial. The one where $t = t_1 t_2$ is immediate by induction as both $[\![\, s :: t_2 \,]\!]$ and $[\![\, s :: t_1 \,]\!]$ have a size smaller than $[\![\, s :: t \,]\!]$. The last case is when t is a variable $x \in V$. Assume that the

variable x appears in an A-production for some $A : \tau = \tau_1 \to \ldots \to \tau_{\varrho(A)} \to \mathrm{o}$ in N. In particular the variable x is of type $\tau_{\mathrm{rk}(x)}$. We have $[\![\, s :: x \,]\!] \stackrel{\text{def}}{=} [\![\, \mathrm{Arg}_{\mathrm{rk}(x)}(s) \,]\!]$. By definition of CStacks, s is an A-stack and using point (1), $\mathrm{Arg}_{\mathrm{rk}(x)}(s)$ is equal to $r :: t'$ with $r \in Stacks$ and $t' : \tau_{\mathrm{rk}(x)} \in \Gamma$. Thus $[\![\, s :: x \,]\!] = [\![\, r :: t' \,]\!]$ for some r smaller than s and using the induction hypothesis, one concludes that $[\![\, s :: x \,]\!]$ is a term in $\mathrm{Terms}_{\tau_{\mathrm{rk}(x)}}(N)$.

(3) Let $s \in \mathrm{WStacks}$ whose top-symbol starts with $\varphi : \tau = \tau_1 \to \ldots \to \tau_{\varrho(\varphi)} \to \mathrm{o}$. Clearly $pop_1(s) :: \varphi$ belongs to CStacks and by point (2), $[\![\, pop_1(s) :: \varphi \,]\!]$ is of type τ. Points (1) and (2) implies that, for all $k \in [1, \varrho(\varphi)]$, $[\![\, \mathrm{Arg}_k(s) \,]\!]$ is of type τ_k. Hence, from Definition 4 it directly follows that $[\![\, s \,]\!]$ is of type o. ∎

3) Proof of Lemma 2:

Lemma 2. *Let $k \in [2, m]$ and let $s = s' +\!\!+ top_k(s) \in \mathrm{WStacks}$. For all non-empty φ-stacks $r \sqsubseteq top_k(s)$, $[\![\, Arg_\ell(s' +\!\!+ r) \,]\!] = [\![\, Arg_\ell(s +\!\!+ r) \,]\!]$ for all $\ell \in [1, \varrho(\varphi)]$.*

Proof: We show, by induction on the size of r, that $s +\!\!+ r$ and $s' +\!\!+ r$ are well-formed and $[\![\, \mathrm{Arg}_\ell(s' +\!\!+ r) \,]\!] = [\![\, \mathrm{Arg}_\ell(s +\!\!+ r) \,]\!]$ for all $\ell \in [1, \varrho(\varphi)]$ where $\varphi \in N \cup V$ denotes the head symbol of $top_1(r)$.

The base case (which considers $[\cdots[\bot Z]_1 \cdots]_k$) is immediate. Assume that the property holds for all substack of $top_k(s)$ smaller than some φ-stack $r \sqsubseteq top_k(s)$. We will show that it holds for r.

The key observation is that: $top_2(s +\!\!+ r) = top_2(s' +\!\!+ r)$ and either $collapse(s +\!\!+ r) = collapse(s +\!\!+ r)$ if the link attached to topmost symbol of r is order greater than k or $collapse(s +\!\!+ r) = s +\!\!+ collapse(r)$ and $collapse(s' +\!\!+ r) = s' +\!\!+ collapse(r)$ otherwise.

As $s' +\!\!+ r$ is a substack of s (which is well-formed), $s' +\!\!+ r$ is well-formed as well. To prove that $s +\!\!+ r$ is well-formed, we need to show that any non-empty substack of $s +\!\!+ r$ satisfies the two properties expressed in Definition 1. The case of a proper substack immediately follows the induction hypothesis. We can deduce that $s +\!\!+ r$ satisfies these two properties from the above observations. Indeed the first property only depends on the top most order-1 stack (and $top_2(s +\!\!+ r) = top_2(s' +\!\!+ r)$) and the second property follows from the fact that $top_1(s +\!\!+ r) = top_1(s' +\!\!+ r)$ and $top_1(collapse(s +\!\!+ r)) = top_1(collapse(s' +\!\!+ r))$.

Assume that the top symbol of r is equal to $\varphi\, t_1 \cdots t_n$. Let $\ell \in [1, \varrho(\varphi)]$ and let us show that $[\![\, \mathrm{Arg}_\ell(s +\!\!+ r) \,]\!] = [\![\, \mathrm{Arg}_\ell(s' +\!\!+ r) \,]\!]$.

⋆ If $\ell \leqslant n$ then $[\![\, \mathrm{Arg}_\ell(s +\!\!+ r) \,]\!] = [\![\, s +\!\!+ pop_1(r) :: t_\ell \,]\!]$ and $[\![\, \mathrm{Arg}_\ell(s +\!\!+ r) \,]\!] = [\![\, s' +\!\!+ pop_1(r) :: t_\ell \,]\!]$. By induction hypothesis, we have that $[\![\, s +\!\!+ r' :: t \,]\!] = [\![\, s' +\!\!+ r' :: t \,]\!]$ for any proper substack r' of r, in particular for $r' = pop_1(r)$.

⋆ If $\ell > n$ then $[\![\, \mathrm{Arg}_\ell(s +\!\!+ r) \,]\!] = [\![\, \mathrm{Arg}_{\ell-n}(collapse(s +\!\!+ r)) \,]\!]$ and $[\![\, \mathrm{Arg}_\ell(s +\!\!+ r) \,]\!] = [\![\, \mathrm{Arg}_{\ell-n}(collapse(s +\!\!+ r)) \,]\!]$. From the above observation, we either have $collapse(s +\!\!+ r) = collapse(s' +\!\!+ r)$ and the equality trivially holds or $collapse(s +\!\!+ r) = s +\!\!+ collapse(r)$ and $collapse(s' +\!\!+ r) = s' +\!\!+ collapse(r)$ in which case the equality follows by induction hypothesis as $|\, collapse(r) \,| < |\, r \,|$. ∎

4) Proof of Lemma 3:

Lemma 3. *Let s be a φ-stack in $\mathrm{WStacks}$ for some $\varphi : \tau_1 \to \ldots \to \tau_{\varrho(\varphi)} \to \mathrm{o}$ in $V \cup N$ and let $\ell \in [1, \varrho(\varphi)]$ with τ_ℓ of order $k > 0$. If $Arg_\ell(s)$ is equal to $r :: t \in \mathrm{CStacks}$ with t starting with $\psi \in N \cup V$ then $pop_{n-k+1}(s) = pop_{n-k+1}(r)$, $|\, top_{n-k+1}(s) \,| > |\, top_{n-k+1}(r) \,|$.*

Proof: We proceed by induction of the size of s. The base case which considers the stack $[\cdots[\,\bot Z\,]_1 \cdots]_n$ is immediate as $\varrho(Z) = 0$.

Assume that the property holds for all stacks in WStacks smaller than some stack $s \in \mathrm{WStacks}$. Let $\varphi\, t_1 \ldots t_m$ be the top symbol of s with $\varphi : \tau_1 \to \ldots \to \tau_{\varrho(\varphi)} \to \mathrm{o}$ in $V \cup N$ and $m \in [0, \varrho(\varphi)]$. Let $\ell \in [1, \varrho(\varphi)]$ and let k be the order of τ_ℓ. Assume that $\mathrm{Arg}_\ell(s) = r :: t$.

If $\ell \leqslant m$, $\mathrm{Arg}_\ell(s) = pop_1 s :: t_\ell$. In particular r is equal to $pop_1(s)$ and the property holds because $pop_{n-k+1}(r) = pop_{n-k+1}(pop_1(s)) = pop_{n-k+1}(s)$ as $n - k + 1 \geqslant 2$ (indeed $k < n$ by definition of n).

If $\ell > m$, $\mathrm{Arg}_\ell(s) = \mathrm{Arg}_{\ell-m}(collapse(s))$. By induction hypothesis, $pop_{n-k+1}(collapse(s)) = pop_{n-k+1}(r)$. To conclude it is enough to show that $pop_{n-k+1}(collapse(s)) = pop_{n-k+1}(s)$. Let k' be the order of $top_1(s)$. As $top_1(s) = \varphi\, t_1 \cdots t_m$ is of type $\tau_m + 1 \to \ldots \to \tau_{\varrho(\varphi)} \to \mathrm{o}$, we have $k' > k$. By definition of well-formed stacks, the order of the link attached to top symbol is equal to $n - k' + 1$. In particular, $pop_{n-k+1}(collapse(s)) = pop_{n-k+1}(s)$. ∎

5) Proof of Proposition 1:

Proposition 1. $\mathrm{Tree}^{\perp}(\mathcal{S}) = \mathrm{Tree}^{\perp}(\mathcal{M})$.

Proof: The proof relies on two lemmas.
The first lemma states the soundness of the first line of the definition of $\xrightarrow[\mathcal{M}]{}$.

Lemma 4. *Let s be an N-stack in* WStacks *and $a \in \Sigma$.*

$$\begin{cases} \exists t \in \mathrm{Terms}(N),\ [\![s]\!] \xrightarrow{a} t & \Rightarrow \quad \exists s' \in \mathrm{WStacks},\ s \xrightarrow[\mathcal{M}]{a} s' \text{ and } [\![s']\!] = t \\ \exists s' \in \mathrm{WStacks},\ s \xrightarrow[\mathcal{M}]{a} s' & \Rightarrow \quad [\![s]\!] \xrightarrow{a} [\![s']\!] \end{cases}$$

Proof: Let $s \in \mathrm{WStacks}$ be an A-stack for some $A \in N$ and let $a \in \Sigma$. By definition of $[\![s]\!]$, $[\![s]\!]$ is equal to $A\,[\![\mathrm{Arg}_1(s)]\!] \cdots [\![\mathrm{Arg}_{\varrho(A)}(s)]\!]$.

Assume that $[\![s]\!] \xrightarrow{a} t$ for some $t \in \mathrm{Terms}(N)$. By definition of $\xrightarrow{a}$, there exists a production $A\,x_1 \cdots x_{\varrho(A)} \xrightarrow{a} t'$ in $\mathcal{R}$ such that t is equal to $t'[x_1/[\![\mathrm{Arg}_1(s)]\!], \ldots, x_{\varrho(A)}/[\![\mathrm{Arg}_{\varrho(A)}(s)]\!]]$. By definition of $\xrightarrow[\mathcal{M}]{a}$, we have $s \xrightarrow[\mathcal{M}]{a} push_1^{t'}(s)$ hence we only need to note that $[\![push_1^{t'}(s)]\!]$ is equal to $t'[x_1/[\![\mathrm{Arg}_1(s)]\!], \ldots, x_{\varrho(A)}/[\![\mathrm{Arg}_{\varrho(A)}(s)]\!]]$. Indeed, as t' is of ground type, $[\![push_1^{t'}(s)]\!]$ is equal to $[\![s :: t']\!]$ which is by definition equal to $t'[x_1/[\![\mathrm{Arg}_1(s)]\!], \ldots, x_{\varrho(A)}/[\![\mathrm{Arg}_{\varrho(A)}(s)]\!]]$.

Now, assume that $s \xrightarrow[\mathcal{M}]{a} s'$ for some $s' \in \mathrm{WStacks}$. By definition of $\xrightarrow[\mathcal{M}]{a}$, there exists a production $A\,x_1 \cdots x_{\varrho(A)} \xrightarrow{a} t' \in \mathcal{R}$ such that $s' = push_1^{t'}(s)$. As s is an A-stack, we have $[\![s]\!] = A\,[\![\mathrm{Arg}_1(s)]\!] \ldots [\![\mathrm{Arg}_{\varrho(A)}(s)]\!]$. Furthermore $[\![s']\!]$ is equal to $t'[x_1/\mathrm{Arg}_1(s), \ldots, x_{\varrho(A)}/\mathrm{Arg}_{\varrho(A)}(s)]$. Hence by definition of $\xrightarrow{a}$, $[\![s]\!] \xrightarrow{a} [\![s']\!]$. ∎

The second lemma states the soundness of the second and third lines of the definition of $\mathcal{M}$. Moreover, it permits to conclude that there are no infinite path labeled by $\boldsymbol{e}$ in $\mathcal{M}$.

Lemma 5. *We have the following properties:*

1) *Let $s \in$ WStacks be a φ-stack for $\varphi \in V$ and let $s' \in$ WStacks be a ψ-stack for $\psi \in V \cup N$. If $s \xrightarrow[\mathcal{M}]{\boldsymbol{e}} s'$ then $[\![s]\!] = [\![s']\!]$, $ord(\varphi) \leqslant ord(\psi)$ and $|\, top_{n-ord(\varphi)+1}(s) \,| > |\, top_{n-ord(\varphi)+1}(s') \,|$.*
2) *For all $s \in$ WStacks there exists a unique N-stack $s' \in$ WStacks such that $s \xrightarrow[\mathcal{M}]{\boldsymbol{e}^*} s'$.*

Proof:
(1) Let φ be a variable in V and let s be a φ-stack in WStacks.We distinguish two cases depending on the order of the φ.
⋆ Assume that φ is of ground type and that $\mathrm{Arg}_{\mathrm{rk}(\varphi)}(pop_1(s))$ is some $r :: t \in$ CStacks.

We have by definition of $\mathcal{M}$ that $s \xrightarrow[\mathcal{M}]{\boldsymbol{e}} s' = push_1^t(r)$. To show that $[\![s]\!]$ is equal to $[\![s']\!]$, we simply unfold the definitions.

$$[\![s]\!] \overset{\text{def}}{=} [\![pop_1(s) :: \varphi]\!] \overset{\text{def}}{=} [\![\mathrm{Arg}_{\mathrm{rk}(\varphi)}(pop_1(s))]\!] \overset{\text{def}}{=} [\![r :: t]\!] \overset{\text{Def 4}}{=} [\![push_1^t(r)]\!] \overset{\text{def}}{=} [\![s']\!]$$

Assume that $s' = push_1^t(r)$ is a ψ-stack for some $\psi \in N \cup V$. We have $ord(\psi) \geqslant ord(\varphi) = 0$. As $|\, \mathrm{Arg}_k(pop_1(s)) \,| \leqslant |\, s \,| - 2$, we have that $|\, top_{n+1}(s) = s \,| > |\, top_{n+1}(s') = s' \,|$.
⋆ Assume that φ is of type $\tau = \tau_1 \to \ldots \to \tau_{\varrho(\varphi)} \to \mathrm{o}$ of order $k > 0$. Assume that $\mathrm{Arg}_{\mathrm{rk}(\varphi)}(pop_1(push_{n-k+1}(s)))$ is equal to $r :: t \in$ CStacks. First recall that, from Lemma 1, we have that $t : \tau$. We have by definition that $s \xrightarrow[\mathcal{M}]{} s' = push_1^{t,n-k+1}(r)$. Let us show that $[\![s]\!] = [\![s']\!]$.
Using Fact 1, we have that:

$$\begin{aligned} [\![s']\!] &= \underbrace{[\![pop_1(s') :: top_1(s')]\!]}_{= [\![pop_1(s) :: \varphi]\!]\ (1)}\ \underbrace{[\![\mathrm{Arg}_1(collapse(s'))]\!]}_{= [\![\mathrm{Arg}_1(s)]\!]\ (2)} \ldots \underbrace{[\![\mathrm{Arg}_{\varrho(\varphi)}(collapse(s'))]\!]}_{= [\![\mathrm{Arg}_{\varrho(\varphi)}(s)]\!]\ (2)} \\ &= [\![pop_1(s) :: \varphi]\!] [\![\mathrm{Arg}_1(s)]\!] \cdots [\![\mathrm{Arg}_{\varrho(\varphi)}(s)]\!] = [\![s]\!]. \end{aligned}$$

The equalities denoted (1) and (2) are proven below:

$$[\![pop_1(s') :: top_1(s')]\!] \overset{\text{def}}{=} [\![r :: t]\!] = [\![\mathrm{Arg}_{\mathrm{rk}(\varphi)}(pop_1(push_{n-k+1}(s)))]\!]$$
$$\overset{\text{Lemma 2}}{=} [\![\mathrm{Arg}_{\mathrm{rk}(\varphi)}(pop_1(s))]\!] = [\![pop_1(s) :: \varphi]\!] \quad (1)$$

and for all $i \in [1, \varrho(\varphi)]$,

$$\begin{array}{rcl} [\![\, \mathrm{Arg}_i(collapse(s'))\,]\!] & = & [\![\, \mathrm{Arg}_i(collapse(push_1^{t,n-k+1}(r)))\,]\!] \\ & = & [\![\, \mathrm{Arg}_i(pop_{n-k+1}(r))\,]\!] \\ & \overset{\text{Lemma 3}}{=} & [\![\, \mathrm{Arg}_i(pop_{n-k+1}(pop_1(push_{n-k+1}(s))))\,]\!] \\ & = & [\![\, \mathrm{Arg}_i(s)\,]\!]. \end{array} \tag{2}$$

As both φ and t have type τ, and as t is of the form $\psi\ t_1 \cdots t_\ell$ for some $\ell \geqslant 0$, it directly follows that $ord(\varphi) \leqslant ord(\psi)$.

The fact that $|\, top_{n-ord(\varphi)+1}(s)\,| > |\, top_{n-ord(\varphi)+1}(s')\,|$ directly follows from Lemma 3.

(2) Assume by contradiction that there exists an infinite sequence $(s_i)_{i\geqslant 0}$ of stacks in WStacks such that for all $i \geqslant 0$, $s_i \xrightarrow[\mathcal{M}]{e} s_{i+1}$. For all $i \geqslant 0$, we denote by t_i the top-symbol of s_i and φ_i the head symbol of t_i. According to (1), the order of the φ_i increases and hence is ultimately constant. Let j and k be such that, for all $i \geqslant j$, $ord(\varphi_i)$ is equal to k. Using (1), the size of the $top_{n-k+1}(s_i)$ is strictly decreasing starting from j which leads the contradiction. ■

By definition of $\mathcal{M}$, only well-formed N-stacks can be the source of non-silent transitions. Let s be a well-formed N-stack. If $[\![\, s\,]\!] \xrightarrow[\mathcal{S}]{a} t$ for $a \in \Sigma$ then the N-stack s' such that $s \xrightarrow[\mathcal{M}]{ae^*} s'$ is such that $[\![\, s'\,]\!] = t$. Conversely if $s \xrightarrow[\mathcal{M}]{ae^*} s'$ for some N-stack s' then $[\![\, s\,]\!] \xrightarrow[\mathcal{S}]{a} [\![\, s'\,]\!]$. ■

6) Proof of Theorem 1:

Theorem 1. *For every labeled recursion scheme $\mathcal{S}$ of order-n, there is an n-CPDA $\mathcal{A}$ that generates the same tree. Moreover, the number of states in $\mathcal{A}$ is linear in the maximal arity appearing in $\mathcal{S}$, and its alphabet is of size linear in the one of $\mathcal{S}$.*

Proof (sktech): Let s be a well-formed stack. We denote by $\langle\!\langle\, s\,\rangle\!\rangle$ the configuration of $\mathcal{A}$ defined by $\langle\!\langle\, s\,\rangle\!\rangle = (q_*, s)$ if s is an N-stack and $\langle\!\langle\, s\,\rangle\!\rangle = (q_{\mathrm{rk}(x)}, s)$ if s is a V-stack whose topmost symbol starts with a variable x.

Clearly for any well-formed N-stack s, $s \xrightarrow[\mathcal{M}]{a} s'$ if and only if $\langle\!\langle\, s\,\rangle\!\rangle \xrightarrow[\mathcal{A}]{a} \langle\!\langle\, s'\,\rangle\!\rangle$.

For any V-stack s, if $s \xrightarrow[\mathcal{M}]{e} s'$ then $\langle\!\langle\, s\,\rangle\!\rangle \xrightarrow[\mathcal{A}]{e^*} \langle\!\langle\, s'\,\rangle\!\rangle$ as intuitively $\xrightarrow[\mathcal{A}]{}$ combines the definition of both $\xrightarrow[\mathcal{M}]{}$ and $\mathrm{Arg}_k(\,\cdot\,)$.

Conversely for all V-stack, if $s \xrightarrow[\mathcal{M}]{e} s'$ and $\langle\!\langle\, s\,\rangle\!\rangle \xrightarrow[\mathcal{A}]{e} \langle\!\langle\, s_2\,\rangle\!\rangle$ then $\langle\!\langle\, s_2\,\rangle\!\rangle \xrightarrow[\mathcal{A}]{e^*} \langle\!\langle\, s'\,\rangle\!\rangle$.

■

C. Proofs Omitted in Section IV

1) Proof of Remark **??***:*

Remark ??. *The second constraint in the definition of Damm-safety can be reformulated as : all argument subterms of an argument subterm of order-k appearing in a right-hand side, have at least order-k.*

Proof: Let us first show that the condition of Definition 7 implies the condition stated in Remark **??**. Let $t : \varphi\, t_1 \cdots t_\ell$ be an argument subterm of a right hand side of a Damm-safe scheme with $\varphi : \tau_1 \to \cdots \to \tau_n \to o$. It is enough to show that the t_i's have at least of order $ord(t)$. If $\ell = 0$ or $\ell = n$ (*i.e.* $ord(t) = 0$), the condition trivially holds. Assume that $0 < \ell < n$. As the scheme is homogeneous, the order of t is $\max_{i\in[\ell+1,n]} ord(\tau_i) + 1 = ord(\tau_{\ell+1}) + 1$. Due to the Damm-safety condition, $ord(\tau_\ell) \geqslant ord(\tau_{\ell+1}) + 1 = t$. Using homogeneity, it implies that $ord(t_i) = ord(\tau_i) \geqslant ord(\tau_\ell) \geqslant ord(t)$.

Let us now show that the condition stated in Remark **??** implies the condition of Definition 7.

Let $t : \varphi\, t_1 \cdots t_k$ be an argument subterm of a right hand side of a Damm-safe scheme with $\varphi : \tau_1 \to \cdots \to \tau_n \to o$ and $0 < k < n$. Using homogeneity, we have $ord(t) = ord(\tau_{k+1}) + 1$. Toward a contradiction, assume that $ord(\tau_k) = ord(\tau_{k+1})$. This would imply that the argument subterm t of order $ord(\tau_{k+1}) + 1 = ord(\tau_k) + 1$ contains the argument subterm t_k of strictly smaller order $ord(\tau_k)$. ■

2) Proof of Proposition 2:

Proposition 2. *Damm-safe schemes are safe and for every safe labeled scheme, there exists a Damm-safe labeled scheme of the same order generating the same tree.*

Proof: Let $\mathcal{S}$ be a Damm-safe scheme, let us show that $\mathcal{S}$ is safe. Assume by contradiction that it is not and let t be an argument subterm of minimal size appearing in a right-hand side and violating the safety condition (*i.e.* t contains a variable of order less than $ord(t)$). The term t can be written $\varphi\, t_1 \cdots t_\ell$ with $\varphi : \tau_1 \to \cdots \to \tau_n \to o$ and $0 < \ell < n$. As $\mathcal{S}$ is Damm-safe, all the t_i's have at least order $ord(t)$ (cf. Remark **??**) and one of them t_{i_0} contains a variable of order strictly smaller than $ord(t)$. This contradicts the minimality of t.

Let $\mathcal{S}$ be a safe scheme. By Corollary 1, we can construct a higher-order pushdown automaton $\mathcal{A}$ of the same order generating the same tree. It is easy to very that the translation of [10] of higher-order pushdown automata into safe schemes in fact produces Damm-safe schemes. Therefore applying it to the automaton $\mathcal{A}$ yields a Damm-safe scheme generating the same tree as $\mathcal{S}$ and of the same order. ■

3) Proof of Theorem 2:

The following proof substantiate the proof sketch given in Section IV only at order 2. We start with the proof at order-2 to get intuition of the objects. Then we sketch the key invariants for the proof in the general case.

Theorem 2 (order-2). *The translation of Section III when applied to a safe* order-2 *recursion scheme yields a link-free CPDA*

Proof: Let $\mathcal{S}$ be an order-2 safe scheme and let $\mathcal{A}$ be the CPDA constructed from $\mathcal{S}$ in Section III-B.

To show that $\mathcal{A}$ is link-free, we first show that, in any reachable configuration[7] $(q, s = [s_1 \dots s_m]_2)$, we can define the target of the links of s using only the symbols appearing in the stack. Then, this stronger result allows us to conclude.

For this, we need to introduce some notations. For $i \in [1, m]$ and $j \in [1, |s_i|]$, we denote by $r(i, j)$, $t(i, j)$ and $o(i, j)$ respectively the j-th symbol of stack s_i, the target (if defined) in $[1, i-1]$ of its link and the order (if defined) of this link. By definition of $\mathcal{A}$, $t(i, j)$ and $o(i, j)$ are defined iff $r(i, j)$ is a term of order $k > 0$ and in this case $o(i, j)$ is equal to $2 - k + 1$

Moreover for $i \in [2, m]$, we let ℓ_i be the smallest index at which s_{i-1} and s_i have a different symbol (or $|s_i| + 1$ if no such index exists).

The stack s satisfies the following properties:

1) for all $i \in [1, |s_1|]$, $t(1, i)$ is undefined;
2) for all $i \in [2, m]$, $\ell_i \leqslant |s_{i-1}|$ and for all $i \in [2, m-1]$, $\ell_i \leqslant |s_i|$;
3) for all $i \in [2, m]$ and $1 \leqslant j < \ell_i$, $t(i, j) = t(i-1, j)$;
4) for all $i \in [2, m]$ with $\ell_i \leqslant |s_i|$, $r(i, \ell_i)$ does not contain a variable of order 0 and is an argument subterm of $r(i-1, \ell_i)$ and if $r(i, \ell_i)$ is of order 1 then $t(i, \ell_i) = i - 1$;

[7] *i.e.* reachable from the initial configuration

5) for all $i \in [2, m]$ with $j \in [\ell_i + 1, |s_i|]$, $t(i, j)$ is undefined;
6) if $m \geqslant 2$ then $\ell_m = |s_m| + 1$ iff $top_1(s) = \varphi t_1 \dots t_h$, $q = q_k$ for some $k \in [1, h]$ such that $ord(t_k) = 1$.

These properties are proved by induction on the length of the shortest path in the LTS from the initial configuration to (q, s) and by inspection of the transitions of $\mathcal{A}$. These properties trivial hold for the initial configuration. Assume that they are verified by a configuration (q, s), let us show that they hold for any configuration (p, r) such that $(q, s) \xrightarrow[\mathcal{P}]{a} (p, r)$.

We distinguish several cases depending on the transition of $\mathcal{A}$ applied to go from (q, s) to (p, s'). Let t be $top_1(s)$ and let ℓ'_i, $t'(i, j)$, $o'(i, j)$, $r'(i, j)$ be the notions corresponding to s'.

⋆ If t starts with $F \in N$ and if $F\, x_1 \cdots x_{\varrho(F)} \xrightarrow{a} e \in \mathcal{R}$.

- **The transition is $\boldsymbol{\delta(q_*, t, a) = (q_*, push_1^e)}$.** Then e starts with a non-terminal and $p = q = q_*$ and $s' = push_1^e(s)$. If $m = 1$, there is nothing to prove. Otherwise, as $q = q_*$, by Property 6, $\ell_m \leqslant |s_m|$ hence, $\ell_m = \ell'_m$. Therefore, all properties for s' are inherited from s. In particular Property 5 is still satisfied as we did not attached a link to the order-0 term we just pushed on top of s.
- **The transition is $\boldsymbol{\delta(q_*, t, a) = (q_{\mathrm{rk}(x)}, id)}$.** Then $s' = s$ and all properties are inherited from s.
- **The transition is $\boldsymbol{\delta(q_*, t, a) = (q_{\mathrm{rk}(x)}, push_1^e; push_2; pop_1)}$.** Then e starts with a variable x of order 1, $p = q_*$, $q = q_{\mathrm{rk}(x)}$ and $s' = pop_1(push_2(push_1^e(s)))$. The most interesting case is that of Property 6. As $\ell'_{m+1} = |s_{m+1}| + 1$, we need to show that $top_1(s') = \varphi t_1 \dots t_h$, $q = q_k$ for some $k \in [1, h]$ such that $ord(t_k) = 1$. The only non-immediate part is that $k \leqslant h$. It holds as otherwise it would imply that $top_1(s')$ is an argument subterm of order-2 (as it misses at least one argument of order-1) which leads a contradiction as the scheme we consider has order-2 (hence all its argument subterms have order $\leqslant 1$).

⋆ If t is a term of the form $\varphi\, t_1 \cdots t_h$ for some $\varphi \in V \cup N$.

- **The transition is $\boldsymbol{\delta(q_k, t, e) = (q_{\mathrm{rk}(t_k)}, pop_1; push_1^{t_k})}$.** Then $k \leqslant h$, $t_k : o$, $p = q_k$, $q = q_{\mathrm{rk}(t_k)}$ and $s' = push_1^{t_k}(pop_1(s))$. If $m = 1$, there is nothing to prove. Otherwise, as $t_k : o$, by Property 6, $\ell_m \leqslant |s_m|$. If $\ell_m < |s_m|$ then all properties are trivially inherited from s. The interesting case is $\ell_m = |s_m|$. In this case, $\ell'_m = |s'_m|$: indeed, Property 4 guaranties that $r(m, \ell_m)$ is an argument subterm of $r(m-1, \ell_m)$, hence t_k is as well an argument subterm of $r(m-1, \ell_m)$ (as it is an argument subterm of $r(m, \ell_m)$) hence differs from $r(m-1, \ell_m)$. It remains to show that Property 4 holds (the others are inherited). As $r'(m, \ell'_m) = t_k$ has order 0, the only thing to prove is that t_k does not contains a variable of order-0. But, as t_k is an argument subterm of t and because t does not contain any variable of order 0 (by induction hypothesis) the same holds for t_k.
- **The transition is $\boldsymbol{\delta(q_k, t, e) = (q_{\mathrm{rk}(t_k)}, pop_1; push_1^{t_k, 2})}$.** Then $k \leqslant h$ and t_k has order 1, $p = q_k$, $q = q_{\mathrm{rk}(t_k)}$ and $s' = push_1^{t_k, n-h+1}(pop_1(s))$. First remark that the previous transition was necessarily $(q_k, push_1^t; push_2; pop_1)$ (coming from state q_*) or $(q_k, collapse)$ (coming from some state q_h with $h > k$). In the first subcase, s_m was a prefix of s_{m-1} hence s'_m and s'_{m-1} differs in their last symbol, meaning that $\ell'_m = |s'_m|$; the only thing to prove is Property 4 holds (the others are inherited) and in particular that t_k does not contain a variable of order 0: but this is a consequence of the **safety constraint** because t_k is an argument subterm of a right-hand-side and as it has order-1 the safety constraint imposes that it does not contain a variable of order 0.
  For the second subcase (the previous transition was a collapse), we remark that $\ell_m \leqslant |s_m|$ (see next case below). Hence we can conclude as in case above when $t_k : o$.
- **The transition is $\boldsymbol{\delta(q_k, t, e) = (q_{k-h}, collapse)}$.** Then $k > h$, $p = q_k$, $q = q_{k-h}$ and $s' = collapse(s)$. All properties except Property 6 are inherited. For Property 6, we use the the second part of Property 2 to crucially guaranty that the stack s' is such that $\ell'_{n'} \leqslant |s'_{n'}|$ (here n' refers to the stack height in s').

We are now ready to conclude that $\mathcal{A}$ is link-free. Inspecting the transitions of $\mathcal{A}$, a $collapse$ operation can only be performed if $q = q_k$ and $top_1(s) = \varphi\, t_1 \dots t_h$ with $k > h$ and $\varphi : (\tau_1, \dots, \tau_m, o)$. Thanks to Definition 1, $\varphi\, t_1 \dots t_h$ is of order-1. Property 5 implies that ℓ_m is either equal to $|s_m|$ or to $|s_m| + 1$. Property 6 implies $\ell_m \neq |s_m| + 1$ as otherwise we would have $k \leqslant h$. Thus, we have $\ell_m = |s_m|$ and by Property 4, $collapse(s) = pop_2(s)$. ∎

To extend the previous proof at any order n, we first need to introduce notations to designate positions in an order-n stack.

Let $s = [s_1, \dots s_m]_n$ be an order-n stack. For $h \in [0, n]$, we inductively define an h-position in s as the index of an order-h stack inside s. An h-position in s is a tuple $\overline{i} \in \mathbb{N}^{n-h}$ such that:

- if $h = n$ there is only one position : the empty tuple ε;
- if $h = n - 1$ then $\overline{i} \in [1, m]$;
- if $h < n - 1$ then $\overline{i} = i\overline{j}$ for some $i \in [1, m]$ and $\overline{j}$ an h-position in s_i.

The set of h-positions in s is denoted $\mathrm{Pos}_h(s)$. For two positions $\overline{i}$ and $\overline{j}$, we write $\overline{i} \leqslant \overline{j}$ if $\overline{i}$ is smaller than $\overline{j}$ for the lexicographic order. We denote by $\max_h(s)$ the maximum element of $\mathrm{Pos}_h(s)$.

For any $h \in [0, n-1]$ and any $\overline{i} \in \mathrm{Pos}_h(s)$, we inductively define the order-h stack occurring at postion $\overline{i}$, denoted $s(\overline{i})$, by:

- if $h = n$ there is only one h-position : the empty tuple ε and we let $s(\varepsilon) = s$;
- if $h = n-1$ then $s(\overline{i}) = s_{\overline{i}}$;
- if $h < n-1$ then as $\overline{i} = i\overline{j}$ with $i \in [1, m]$ and $\overline{j}$ is an h-position in s_i, we take $s(\overline{i}) = s_i(\overline{j})$.

Notation 3. *For a symbol appearing in the stack x, $R(x)$, $T(x)$ and $O(x)$ respectively designate the symbol (in Γ), the target of the link (if defined) and the order of the link (if defined) appearing in x.*

By construction of $\mathcal{A}$, $T(s(\overline{i}))$ and $O(s(\overline{i}))$ are defined iff $R(s(\overline{i}))$ is a term of order $k > 0$ and in this case $O(S(\overline{i}))$ is equal to $n - k + 1$.

For all $h \in [0, n-1]$ and all $\overline{i} = (i_1, \ldots, i_{|\overline{i}|}) \in \mathrm{Pos}_h(s)$, we define $\mathrm{pred}(\overline{i}) \in \mathrm{Pos}_h(s)$ and $\sup(\overline{i}) \in \mathrm{Pos}_{h+1}(s)$ if $h < n$ by $\mathrm{pred}(\overline{i}) = (i_1, \ldots, i_{|\overline{i}|} - 1)$ if $i_{|\overline{i}|} > 1$ and $\mathrm{pred}(\overline{i})$ is undefined otherwise and $\sup(\overline{i}) = (i_1, \ldots, i_{|\overline{i}|-1})$.

For all $h \in [0, n-1]$, the position $\overline{i} = (i_1, \ldots, i_{|\overline{i}|}) \in \mathrm{Pos}_h(s)$ is initial if $i_{|\overline{i}|} = 1$. In other terms, $\overline{i}$ is initial if and only if $\mathrm{pred}(\overline{i})$ is undefined.

For all $h \in [1, n-1]$, we define a partial mapping ℓ^h_s with domain in $\mathrm{Pos}_h(s)$ associating to an h-position $\overline{i}$ a 0-position in $s(\overline{i})$. The value of $\ell^h_s(\overline{i})$ (when it is defined) is the smallest 0-position $\overline{j}$ in $s(\overline{i})$ such that $r(\overline{i})$ differs from $R(\prec(\overline{i}))$. Remark that $\ell^h_s(\overline{i})$ is undefined for a non-initial $\overline{i}$ if and only if $s(\overline{i})$ is a substack of $s(\mathrm{pred}(\overline{i}))$.

Theorem 2 (arbitrary order). *The translation of Section III when applied to a safe* order-n *recursion scheme yields a link-free CPDA*

Proof: We define by induction on the order h the notion of safe stack. All order-1 stacks are safe stacks. An oder-$(h+1)$ stack $s = [s_1 \ldots s_m]_{h+1}$ is safe if:

1) for all $i \in [1, m]$, s_i is a safe order-h stack.
2) for all $i \in [2, m]$, $\ell^h_s(i)$ is defined and is such that $\ell^h_s(i)$ belongs to $\mathrm{Pos}_0(s_{i-1})$ and $R(s_i(\ell^h_s(i)))$ is an argument subterm of $R(s_{i-1}(\ell^h_s(i)))$ which does not contain any variable of order $< n - h + 1$. If $R(s_i(\ell^h_s(i)))$ is of order > 0 then $T(s_i(\ell^h_s(i))) = i - 1$. For all $\overline{j} \geqslant \ell^h_s(i)$ in $\mathrm{Pos}_0(s_i)$, $R(s_i(\overline{j}))$ is not of order $n - h + 1$.

We slightly relax the notion of safe stack by defining w-safe stack for $w \in [2, h+1]$. An oder-$(h+1)$ stack $s = [s_1 \ldots s_m]_{h+1}$ is w-safe if:

1) for all $i \in [1, m-1]$, s_i is a safe order-h stack, s_m is safe if $w = h + 1$ and s_m is w-safe otherwise;
2) for all $i \in [2, m']$ where $m' = m$ if $w < h + 1$ and $m' = m - 1$ otherwise, $\ell^h_s(i)$ is defined and is such that $\ell^h_s(i)$ belongs to $\mathrm{Pos}_0(s_{i-1})$ and $R(s_i(\ell^h_s(i))$ is an argument subterm of $R(s_{i-1}(\ell^h_s(i)))$ which does not contain any variable of order $< n - h + 1$. If $R(s_i(\ell^h_s(i)))$ is of order > 0 then $T(s_i(\ell^h_s(i))) = i - 1$. For all $\overline{j} \geqslant \ell^h_s(i)$ in $\mathrm{Pos}_0(s_i)$, $R(s_i(\overline{j}))$ is not of order $n - h + 1$.

By an induction similar to the order-2 case, we can prove that for all reachable configuration (q, s), s is w-safe iff $top_1(s)$ starts with $\varphi : (\tau_1, \ldots, \tau_{\varrho(\varphi)}, \mathrm{o})$, $q = q_i$ with $ord(\tau_i) = n - w + 1$; otherwise s is safe.

We are now ready to conclude that $\mathcal{A}$ is link-free. Inspecting the transitions of $\mathcal{A}$, a $collapse$ operation can only be performed if $q = q_k$ and $top_1(s) = \varphi\, t_1 \ldots t_h$ with $k > h$ and $\varphi : (\tau_1, \ldots, \tau_m, \mathrm{o})$. Let us write $w = n - ord(\tau_k) + 1$. By the above property s is w-safe. Let $w' = n - ord(top_1(s)) + 1$. We have $w' < w$ and the link is of order w'. Let s' be the topmost order-w' stack of s. By definition of w-safety it is safe. The last part of Property 2 implies that $\ell^{w'}_{s'}$ is $\max_{w'}(s')$ hence Property 2 implies that the link of the top-most symbol of s' point to the previous order-$(w'-1)$ stack. Hence $collapse(s) = pop_{w'}(s)$.

■

D. Collapsible Pushdown Games: Complexity and Winning Strategies

This section is devoted to the study of parity games played on LTS of CPDA. For simplicity of presentation we omit the input alphabet and introduce the concept of Collapsible pushdown processes (CPDP). The main focus is on complexity of deciding the winner and in establishing the existence of strategies realised by CPDA transducer synchronised with the one defining the game. This latter result is the key ingredient to prove the effective selection property.

Comparison with previous proofs from [9] and [2].

The lines behind the proof of the main result of this section (Theorem 5) slightly differ from the one of a similar statement (where complexity was not studied precisely, neither strategies) from [9]. Indeed, in [9] the induction step removes the outmost links at the same time as in reduces the order. As a consequence, the definition of collapse rank was different, and the transformation from a usual CPDA to a rank-aware one is different. Separating both steps (removing the link and decreasing the order) seemed necessary with respect to designing strategies realised by CPDA, which is crucially used later for effective selection. Note that a similar proof technique (but without any consideration on strategies and precise complexity) was considered in [2].
Hence the main added value here are: precise (and improved) analysis of the complexity (which requires several optimisations) and existence of winning strategies realised by CPDA synchronised with the one defining the game.

1) Collapsible Pushdown Processes:

First we introduce, for any stack symbol γ an operation on stacks that does top_1 rewriting: this operation, denoted rew_1^γ, takes the top_1 element and replace it by γ without modifying the link. Formally:

$$rew_1^\gamma \underbrace{[\, s_1 \cdots s_{l+1} \,]}_{s} = \begin{cases} [\, s_1 \cdots s_l \, rew_1^\gamma s_{l+1} \,] & \text{if } ords > 1 \\ [\, s_1 \cdots s_l \, \gamma^{(j,k)} \,] & \text{if } ords = 1 \text{ and } s_{l+1} = \alpha^{(j,k)} \end{cases}$$

We also let id for the identity operation (*i.e.* $id(s) = s$ for all stack s).

We now introduce the notion of collapsible pushdown processes which differs from CPDA from the fact that they have no input alphabet (and can be non-deterministic).

An *order-n collapsible pushdown process* (*n-CPDP*) is a 4-tuple $\mathcal{A} = \langle\, \Gamma, Q, \Delta, q_0 \,\rangle$ where Γ is the stack alphabet, Q is the finite set of control states, $q_0 \in Q$ is the initial state, and $\Delta : Q \times \Gamma \to 2^{Q \times Op_n(\Gamma) \times Op_n(\Gamma)}$ is the transition function and satisfies the following constraint. For any $q, \gamma \in Q \times \Gamma$, for any $(q', op_1, op_2) \in \Delta(q, \gamma)$ one has that $op_1 \in \{rew_1^\alpha \mid \alpha \in \Gamma\} \cup \{id\}$ and $op_2 \notin \{rew_1^\alpha \mid \alpha \in \Gamma\}$: hence a transition will always act on the stack by doing (possibly) rewriting the top symbol and then (possibly) doing another kind of operation on the stack. In the following we will use notation $(q', op_1; op_2)$ instead of (q', op_1, op_2) (to stress that one performs op_1 followed by op_2).

Configurations of an n-CPDP are pairs of the form (q, s) where $q \in Q$ and s is an n-stack over Γ; we call $(q_0, \bot_n)$ the *initial configuration*, where $\bot_n = [\ldots [\, \bot \,]_1 \ldots]_n$.

An n-CPDP $\mathcal{A} = \langle\, \Gamma, Q, \Delta, q_0 \,\rangle$ naturally defines a transition graph $\mathrm{Graph}(\mathcal{A}) := (V, E \subseteq V \times V)$ whose vertices V are the configurations of $\mathcal{A}$ and whose edge relation E is given by: $((q, s), (q', s')) \in E$ iff $\exists (q', op_1; op_2) \in \Delta(q, top_1(s))$ such that $s' = op_2(op_1(s))$. Such a graph is called an *n-CPDP graph*.

2) n-CPDP Parity Games:

Let $G = (V, E \subseteq V \times V)$ be a graph. Let $V_{\mathbf{E}} \uplus V_{\mathbf{A}}$ be a partition of V between two players, Éloïse and Abelard. A *arena* is such a tuple $\mathcal{G} = (G, V_{\mathbf{E}}, V_{\mathbf{A}})$. A colouring function Ω is a mapping $\Omega : V \longrightarrow C \subset \mathbb{N}$ where C is a finite set of colours. An *infinite two-player parity game* on an arena $\mathcal{G}$ is a pair $\mathbb{G} = (\mathcal{G}, \Omega)$.

Éloïse and Abelard play in $\mathbb{G}$ by moving a pebble between vertices. A *play* from some initial vertex v_0 proceeds as follows: the player owning v_0 moves the pebble to a vertex v_1 such that $(v_0, v_1) \in E$. Then the player owning v_1 chooses a successor v_2 and so on. If at some point one of the players cannot move, she/he loses the play. Otherwise, the play is an infinite word $v_0 v_1 v_2 \cdots \in V^\omega$ and is won by Éloïse just in case $\liminf(\Omega(v_i))_{i \geqslant 0}$ is even. A *partial play* is just a prefix of a play.

A *strategy* for Éloïse is a function assigning, to every partial play ending in some vertex $v \in V_{\mathbf{E}}$, a vertex v' such that $(v, v') \in E$. Éloïse *respects a strategy* Φ during a play $\Lambda = v_0 v_1 v_2 \cdots$ if $v_{i+1} = \Phi(v_0 \cdots v_i)$, for all $i \geqslant 0$ such that $v_i \in V_{\mathbf{E}}$.

A strategy Φ for Éloïse is *winning* from a position $v \in V$ if she wins every play that starts from v and respects Φ. Finally, a vertex $v \in V$ is *winning* for Éloïse if she has a winning strategy from v. Symmetrically, one defines the corresponding notions for Abelard. It follows from Martin's determinacy Theorem [A5] that, from every position, either Éloïse or Abelard has a winning strategy.

Now let $\mathcal{A} = \langle\, \Gamma, Q, \Delta, q_0 \,\rangle$ be an order-n CPDP and let $\mathrm{Graph}(\mathcal{A}) = (V, E)$ be its transition graph. Let $Q_{\mathbf{E}} \uplus Q_{\mathbf{A}}$ be a partition of Q and let $\Omega : Q \longrightarrow C \subset \mathbb{N}$ be a colouring function (over states). Altogether they define a partition $V_{\mathbf{E}} \uplus V_{\mathbf{A}}$ of V whereby a vertex belongs to $V_{\mathbf{E}}$ iff its control state belongs to $Q_{\mathbf{E}}$, and a colouring function $\Omega : V \longrightarrow C$ where a vertex is assigned the colour of its control state. The structure $\mathcal{G} = (\mathrm{Graph}(\mathcal{A}), V_{\mathbf{E}}, V_{\mathbf{A}})$ defines an arena and the pair $\mathbb{G} = (\mathcal{G}, \Omega)$ defines a parity game (that we call a *n-CPDP parity game*).

Given an n-CPDP parity game, we will consider the following two algorithmic questions:

1) Decide whether $(q_0, \bot_n)$ is winning for Éloïse.
2) If $(q_0, \bot_n)$ is winning for Éloïse, provide a description of a winning strategy for Éloïse from $(q_0, \bot_n)$.

To answer the second question we will consider *strategies realised by n-CPDA transducers*.

3) CPDA strategies:

Let $\mathcal{A} = \langle\, \Gamma, Q, \Delta, q_0 \,\rangle$ be an order-n CPDP, let $\mathrm{Graph}(\mathcal{A}) = (V, E)$ be its transition graph, let $\mathcal{G} = (\mathrm{Graph}(\mathcal{A}), V_{\mathbf{E}}, V_{\mathbf{A}})$ be an arena associated with $\mathcal{A}$ and let $\mathbb{G} = (\mathcal{G}, \Omega)$ be a corresponding n-CPDP parity game.

We aim at defining a notion of n-CPDA *transducers* that provide a description for strategies in $\mathbb{G}$, that is describe a function from partial plays in $\mathbb{G}$ into V.

Consider a partial play $\Lambda = v_0 v_1 \cdots v_\ell$ in $\mathbb{G}$ where $v_0 = (q_0, \bot_n)$. An alternative description of Λ is by the sequence $(q_1, rew_1; op_1) \cdots (q_\ell, rew_\ell; op_\ell) \in (Q \times Op_n(\Gamma) \times Op_n(\Gamma))^*$ such that $v_i = (q_i, s_i)$ for all $1 \leqslant i \leqslant \ell$ and $s_i = op_i(rew_i(s_{i-1}))$ (with the convention that $s_0 = \bot_n$). We may in the following use implicitly this representation of Λ when needed. Similarly, one can represent a strategy as a (partial) function $\Phi : (Q \times Op_n(\Gamma) \times Op_n(\Gamma))^* \to Q \times Op_n(\Gamma) \times Op_n(\Gamma)$, the meaning being that in a partial play Λ ending in some vertex (q, s) if $\Phi(\Lambda) = (q', rew; op)$ then the player moves to $(q', op(rew(s)))$.

An *n-CPDA transducer realising a strategy* in $\mathbb{G}$ is a tuple $\mathcal{S} = \langle\, \Sigma, S, \delta, \tau, s_0 \,\rangle$ where Σ is a stack alphabet, S is a finite set of states, $s_0 \in S$ is the initial state,

$$\delta : S \times \Sigma \times (Q \times Op_n(\Gamma) \times Op_n(\Gamma)) \to S \times Op_n(\Sigma) \times Op_n(\Sigma)$$

is a *deterministic* transition function and

$$\tau : S \times \Sigma \to Q \times Op_n(\Gamma) \times Op_n(\Gamma)$$

is a deterministic choice function (note that we do not require τ to be total). For both δ and τ we do the same requirement as for the transition function for CPDP, namely that the first stack operation should be a top-rewriting (or the identity) and that the second one should not be a top-rewriting.

A configuration of $\mathcal{T}$ is a pair (s, σ) where s is a state and σ is an n-stack over Σ; the initial configuration of $\mathcal{T}$ is $(s_0, \bot_n)$. With a configuration (s, σ) is associated, when defined, a (unique) move in $\mathbb{G}$ given by $\tau(s, top_1(\sigma))$ A partial play $\Lambda = (q_1, rew_1, op_1) \cdots (q_\ell, rew_\ell, op_\ell)$ in $\mathbb{G}$ induces a (unique, when defined) *run* of $\mathcal{T}$ which is the sequence such that

$$(s_0, \sigma_0)(s_1, \sigma_1) \cdots (s_\ell, \sigma_\ell)$$

where $(s_0, \sigma_0) = (s_0, \bot_n)$ is the initial configuration of $\mathcal{T}$ and for all $0 \leqslant i \leqslant \ell - 1$ one has $\delta(s_i, top_1(\sigma_i), (q_{i+1}, rew_{i+1}; op_{i+1})) = (s_{i+1}, rew'_{i+1}; op'_{i+1})$ with $\sigma_{i+1} = op'_{i+1}(rew'_{i+1}(\sigma_i))$. In other words, the control state and the stack of $\mathcal{T}$ are updated accordingly to δ.

We say that $\mathcal{T}$ is *synchronised* with $\mathcal{A}$ iff for all $(s, a, (q, rew; op)) \in S \times \Sigma \times (Q \times Op_n(\Gamma) \times Op_n(\Gamma))$ such that $\delta(s, a, (q, rew; op)) = (s', rew'; op')$ is defined one has that op and op' are of the same kind, *i.e.* either they are both a pop_k (for some k) or both a $push_k$ (for some k) or both a $push_1^{,e}$ (the symbol pushed being possibly different but the order of the link being the same) or both $collapse$ or both id. In particular, if one defines the *shape* of a stack s as the stack obtained by replacing all symbols appearing in s by a fresh symbol $\sharp$ (but keeping the links) one has the following.

Proposition 3. *Assume that $\mathcal{T}$ is synchronised with $\mathcal{A}$. Then, for any partial play Λ in $\mathbb{G}$ ending in a configuration with stack s, the run of $\mathcal{T}$ on Λ, when exists, ends in a configuration with stack σ such that s and σ have the same shape.*

The *strategy realised by* $\mathcal{T}$ is the (partial) function $\varphi_{\mathcal{S}}$ defined by letting $\varphi_{\mathcal{S}}(\Lambda) = \tau((s, top_1(\sigma)))$ where (s, σ) is the last configuration of the run of $\mathcal{T}$ on Λ.

We say that Éloïse *respects* $\varphi_{\mathcal{S}}$ during a partial play $\Lambda = (q_1, rew_1; op_1) \cdots (q_\ell, rew_\ell; op_\ell)$ in $\mathbb{G}$ iff for all $0 \leqslant i \leqslant \ell - 1$ if the last configuration in $(q_1, rew_1; op_1) \cdots (q_i, rew_i; op_i)$ belongs to $V_{\mathbf{E}}$ then $(q_{i+1}, rew_{i+1}; op_{i+1}) = \varphi_{\mathcal{S}}((q_1, rew_1; op_1) \cdots (q_i, rew_i; op_i))$

We say that $\varphi_{\mathcal{S}}$ is *well-defined* iff for any partial play $\Lambda = (q_1, rew_1; op_1) \cdots (q_\ell, rew_\ell; op_\ell)$ where Éloïse respects $\varphi_{\mathcal{S}}$ if the last vertex (q_ℓ, s_ℓ) in Λ belongs to $V_{\mathbf{E}}$ then $\varphi_{\mathcal{S}}(\Lambda) \in \Delta(q, top_1(s_\ell))$, *i.e.* the move given by $\varphi_{\mathcal{S}}$ is a valid one.

4) Main Result:

Theorem 5. *Let $\mathcal{A} = \langle \Gamma, Q, \delta, q_0 \rangle$ be an n-CPDA and let $\mathbb{G}$ be an n-CPDA parity game defined from $\mathcal{A}$. Then one has the following results.*

1) *Deciding whether $(q_0, \bot_n)$ is winning for Éloïse is an n-*EXPTIME *complete problem.*
2) *If $(q_0, \bot_n)$ is winning for Éloïse then one can effectively construct an n-CPDA transducer $\mathcal{T}$* synchronised *with $\mathcal{A}$ realising a well-defined* winning *strategy for Éloïse in $\mathbb{G}$ from $(q_0, \bot_n)$.*

The proof is by induction on the order and each induction step is itself divided into two steps: the first one removes the outermost links while the second one lowers the order.

Before going to the proof, we give in Section D5 a normalisation result (Theorem 6). Then Section D6 explains how to removes the outermost links and Section D7 shows how to reduce the order. Finally Section D8 combines the previous constructions and provides the proof of Theorem 5 together with a precise complexity analysis.

5) Rank-aware CPDP:

Fix, for the whole subsection, an n-CPDP $\mathcal{A} = \langle \Gamma, Q, \Delta, q_0 \rangle$, a partition $Q_{\mathbf{E}} \uplus Q_{\mathbf{A}}$ of Q and a colouring function $\Omega : Q \to C \subset \mathbb{N}$. Denote by G its transition graph, by $\mathcal{G}$ the arena induced by G and the partition $Q_{\mathbf{E}} \uplus Q_{\mathbf{A}}$ and by $\mathbb{G}$ the parity game $(\mathcal{G}, \Omega)$.

Let s be an order-n stack. We first associate with $s = s_1, \cdots, s_\ell$ a well-bracketed word of depth n, $\widetilde{s} \in (\Sigma \cup \{ [\, ,] \})^*$:

$$\widetilde{s} := \begin{cases} [\, \widetilde{s_1} \cdots \widetilde{s_\ell} \,] & \text{if } n \geqslant 1 \\ s & \text{if } n = 0 \text{ (\textit{i.e.} } s \in \Sigma) \end{cases}$$

In order to reflect the link structure, we define a partial function $target(s) : \{1, \cdots, |\widetilde{s}|\} \to \{1, \cdots, |\widetilde{s}|\}$ that assigns to every position in $\{1, \cdots, |\widetilde{s}|\}$ the index of the end of the stack targeted by the corresponding link (if exists; indeed this is undefined for $\bot$, $[$ and $]$). Thus with s is associated the pair $\langle \widetilde{s}, target(s) \rangle$; and with a set S of stacks is associated the set $\widetilde{S} = \{\langle \widetilde{s}, target(s) \rangle \mid s \in S\}$.

Example 16. Consider the stack $s = [\,[\,[\,\bot\,\alpha\,]\,]\,[\,[\,\bot\,]\,[\,\bot\,a\,\beta\,\gamma\,]\,]\,]$. Then $\widetilde{s} = [\,[\,[\,\bot\,\alpha\,]\,]\,[\,[\,\bot\,]\,[\,\bot\,\alpha\,\beta\,\gamma\,]\,]\,]$ and $target(5) = 4$, $target(14) = 13$, $target(15) = 11$ and $target(16) = 7$.

A *finite path* in G is a non-empty sequence of configurations $v_0 v_1 \cdots v_m$ such that for all $0 \leqslant i \leqslant m - 1$, there is an edge in G from v_i to v_{i+1}. An *infinite path* is an infinite sequence of configurations $v_0 v_1 \cdots$ such that for all $i \geqslant 0$, there is an edge in G from v_i to v_{i+1}. Note that we do not require v_0 to be the initial configuration.

We now define a generalisation of n-stacks called *indexed n-stacks*. Recall that a stack s is equivalently described as a pair $\langle \widetilde{s}, target(s) \rangle$ (recall that $\widetilde{s}$ is a well-bracketed word description of s and that $target(s)$ gives the link structure). An *indexed n-stack* is described by a triple $\langle \widetilde{s}, target(s), ind(s) \rangle$ where $\widetilde{s} = \widetilde{s}_1 \cdots \widetilde{s}_{|\widetilde{s}|}$ and $target(s)$ are as previously and where $ind(s) : \{1, \ldots, |\widetilde{s}|\} \to \mathbb{N}$ is a partial function that is defined in any position $j < |\widetilde{s}| - n$ such that $\widetilde{s}_j \notin \{[\, ,]\}$. The previous conditions on the domain of $ind(s)$ ensure that any symbol in s which is not the topmost one has a value by $ind(s)$ that we refer to as its *index*. An *indexed configuration* is a pair formed by a control state and an indexed stack.

The *erasure* of an indexed n-stack $\langle \widetilde{s}, target(s), ind(s) \rangle$ is the n-stack $\langle \widetilde{s}, target(s) \rangle$. We extend the notion of erasure to indexed configuration in the obvious way.

With any path $\Lambda = v_0 v_1 \cdots$, with $v_i = (p_i, s_i)$ for all $i \geqslant 0$, we inductively associate a sequence of indexed configurations $\Lambda' = v'_0 v'_1 \cdots$ such that the following holds.

- The erasure of Λ' equals Λ (the *erasure* of a sequence of indexed configurations being defined as the sequence of the respective erasures).
- For any indexed configuration $v'_m = (q_m, s'_m)$ the following holds. Denote by $s'_m = \langle \widetilde{s'_m}, target(s'_m), ind(s'_m) \rangle$, let $\widetilde{s'_m} = x_1 \cdots x_h$, and let j be in the domain of $ind(s'_m)$ such that $x_{j+1} =]$. Then let $j' > j$ be the largest integer such

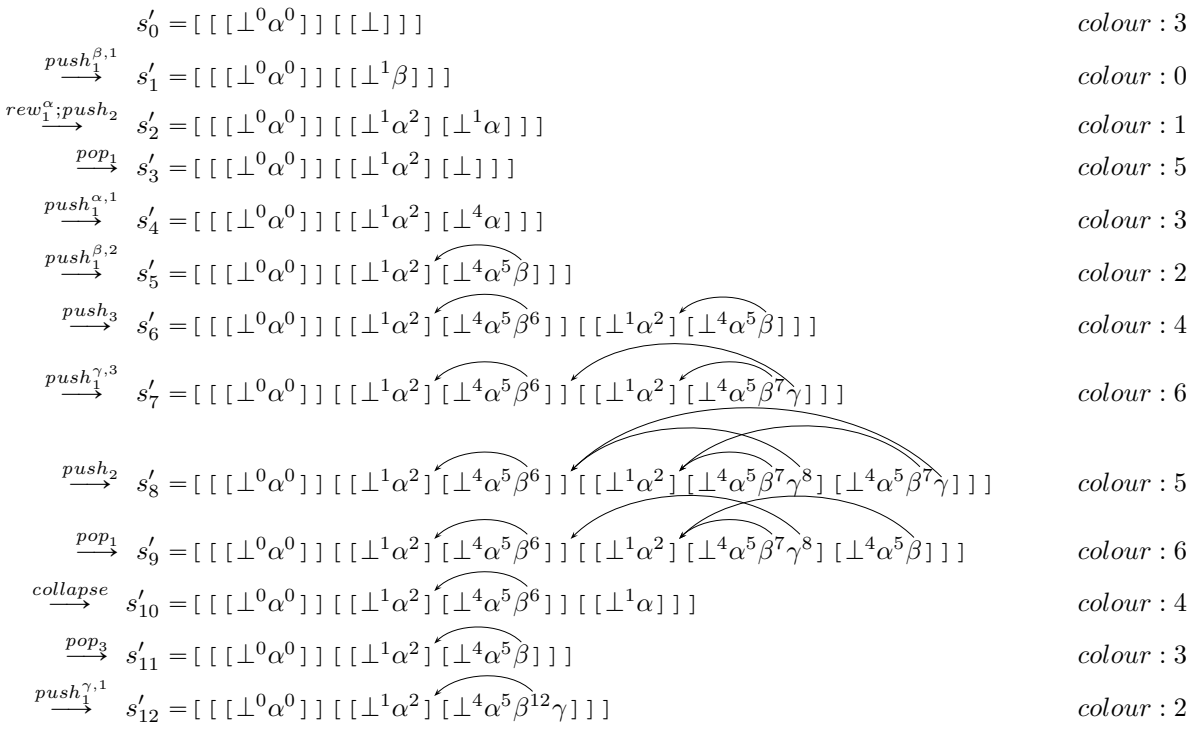


Figure 4. Example of a sequence of indexed stacks

that $x_k =]$ for all $j+1 \leqslant k \leqslant j'$ and let i be the unique integer such that $x_i \cdots x_{j'}$ is well-bracketed. Then, for any $i < k < j'$, if $ind(s'_m)(k)$ is defined, one has $ind(s'_m)(k) \leqslant ind(s'_m)(j)$, and this inequality is strict if $ind(s'_m)(j) \neq 0$. Intuitively, position j is the topmost symbol of some $(j'-j)$-stack, and any symbol in this stack has an index smaller than the topmost symbol.

The intended meaning of the index of some symbol in the stack is the following. The index is equal to the largest integer i such that since v_i/v'_i the symbol no longer appears as a top_1-element. If one uses the stack to store (and maintain) some information, the index is the moment from which this information was no longer updated. Hence when some symbol appears again as the top_1-element, one has to update the information by taking into account all that happened since v_i/v'_i (included).

The intuitive idea behind the forthcoming definition of Λ' is rather simple. The indices are always preserved, so one only cares on new positions in the stack. On doing a $push_k$ the indices of the copied stack are inherited from the original copy. Then when new indices are needed (because a position is no longer the top_1 one, it get index $m+1$ if the current configuration is v_{m+1}).

Before going to the formal definition, we start with an example.

Example 17. *In figure 4, we give an example (at order 3) that illustrates the previous intuitive idea as well as the formal description below (ignore the information on colours for this example). We only describe the indexed stacked (omitting the control states), and indicate the stack operation (but omit the id operation). Indices are written as superscripts.*

Now, we formally give the construction (the previously mentioned properties easily follow from the definition). The initial configuration $v'_0 = (p_0, s'_0)$, is obtained by letting $ind(s'_0)$ be the constant (partial) function equal to 0. Assume now that $v'_1 \cdots v'_m$ has been constructed, let $v'_m = (p_m, s'_m)$ with $s'_m = \langle\, \widetilde{s}_m, target(s_m), ind(s'_m) \,\rangle$ and let $v_{m+1} = (p_{m+1}, s_{m+1})$ with $s_{m+1} = \langle\, \widetilde{s}_{m+1}, target(s_{m+1}) \,\rangle$. We let $v'_{m+1} = (p_{m+1}, s'_{m+1})$ with $s'_{m+1} = \langle\, \widetilde{s}_{m+1}, target(s_{m+1}), ind(s'_{m+1}) \,\rangle$ where $ind(s'_{m+1})$ is defined thanks to a case distinction.

- A top-rewriting operation followed by a $push_1^{\gamma,k}$ operation is applied in configuration v_m. Then all previous indices are inherited and the former top_1-element gets index $m+1$. Formally, $ind(s'_{m+1})(j) = ind(s'_m)(j)$ whenever $j < |\widetilde{s}_m| - n$ and $ind(s'_{m+1})(|\widetilde{s}_m| - n) = m+1$.

- A top-rewriting operation followed by a $push_k$ operation is applied: $s_{m+1} = push_k(s_m)$. First, all existing indices are preserved, *i.e.* $ind(s'_{m+1})(j) = ind(s'_m)(j)$ whenever j belongs to the domain of $ind(s'_m)$. Then one writes $\widetilde{s}_m$ as $[\cdots[t]]^{n-k+1}$ with t being well-bracketed; hence, $\widetilde{s}_{m+1} = [\cdots[t][t]]^{n-k+1}$. Then we let $ind(s'_{m+1})(|\widetilde{s'_m}| - (n-k+1)+j) = ind(s'_m)(|\widetilde{s'_m}| - (n-k+1) - (|t|+2) + j)$ for all $j \geqslant 1$ such that the second member of the equality is defined: the indices are simply copied from the former top $(k-1)$-stack. Finally, the former top_1-element gets index $m+1$: $ind(s'_{m+1})(|\widetilde{s}_m| - n + k - 3) = m+1$.
- A top-rewriting operation followed by a either a pop_k operation or a $collapse$ or id is applied in configuration v_m in Λ. Then all indices are inherited from the previous indexed stack. Formally, $ind(s'_{m+1})(j) = ind(s'_m)(j)$ whenever j belongs to the domain of $ind(s'_{m+1})$.

The following proposition is crucial for the rest of the proof. In particular, it means that if we stored some information on the stack, the index gives the "expiry date" of the stored information, that is the step in the computation starting from which the information has no longer been updated.

Proposition 4. *Let $\Lambda = v_0 v_1 \cdots$ be a path and $\Lambda' = v'_0 v'_1 \cdots$ be as above. Let $m \geqslant 0$, let $s'_m = \langle\, \widetilde{s}_m, target(s_m), ind(s'_m) \,\rangle$ be the indexed stack in v'_m. Let j be such that $i = ind(s'_m)(j)$ is defined. If $i > 0$, then $(i-1)$ is the largest integer such that the j-th letter of $\widetilde{s}_m$ is a copy of $top_1(s_{i-1})$. If $i = 0$, there is not i' such that the j-th letter of $\widetilde{s}_m$ is a copy of $top_1(s_{i'})$.*

Proof: Immediate by induction on m and from the definition of Λ' from Λ. ∎

Our main goal is to enrich the stack alphabet in order to compute the *link-rank*. Assume that in configuration v_m the top_1-element has a link (that is possibly a copy of a link) that was created in configuration v_j: then the link-rank in v_m is defined as the smallest colour since the creation of the link, *i.e.* $\min\{\Omega(v_j), \cdots \Omega(v_m)\}$. In order to maintain this information, we need to define two other concepts: the collapse-rank (for updating after performing a $collapse$) and the pop-rank for k (for updating after performing a pop_k).

We first introduce the notion of *ancestor*. Fix a finite path $\Lambda = v_0 v_1 \cdots v_m$, let $v_m = (q, s)$ be some configuration in Λ and let x be a symbol in s. Then the *ancestor* of x is the configuration v_i where i is the index of x in v'_m (the indexed version of v_m).

We now introduce the notion of *collapse-rank*. Fix a finite path $\Lambda = v_0 v_1 \cdots v_m$ and assume that the top_1-element of v_m has a $(k+1)$-link for some k. Then the *collapse-ancestor* in v_m is the ancestor of the top_1-element of the pointed k-stack and the *collapse-rank* in v_m is the smallest colour visited since the collapse-ancestor (included).

Example 18. *Consider the sequence of indexed stacks given in Figure 4 (the colours of the corresponding configurations are indicated on the right part of the figure).*

In v'_8 the collapse-ancestor is 6 *and the collapse-rank is therefore* 4*. In v'_9 the collapse-ancestor is* 2 *and the collapse-rank is therefore* 1*.*

Next, we give a notion of *pop-rank*. Fix a partial play $\Lambda = v_0 v_1 \cdots v_m$ and a configuration $v_m = (q, s)$ in Λ. Then, for any $1 \leqslant k \leqslant n$, the *pop-ancestor* for k, when defined, is the ancestor of the top_1-element of $pop_k(s)$ and the *pop-rank* for k, when defined, is the smallest colour visited since the pop-ancestor for k (included). In particular, the pop-rank for n is the smallest colour visited since the stack has height at least the height of s.

Example 19. *Again, consider the sequence of indexed stacks given in Figure 4.*

*In configuration v'_9 the pop-ancestor (*resp. *pop-rank) for* 3 *is* 6 *(*resp. 4*), the pop-ancestor (*resp. *pop-rank) for* 2 *is* 8 *(*resp. 5*) and the pop-ancestor (*resp. *pop-rank) for* 1 *is* 5 *(*resp. 2*).*

*In configuration v'_{12} the pop-ancestor (*resp. *pop-rank) for* 3 *is* 0 *(*resp. 0*), the pop-ancestor (*resp. *pop-rank) for* 2 *is* 2 *(*resp. 1*) and the pop-ancestor (*resp. *pop-rank) for* 1 *is* 12 *(*resp. 2*).*

Consider a finite path $\Lambda = v_0 v_1 \cdots v_m$ in $\mathbb{G}$ ending in a configuration $v_m = (q, s)$ such that $top_1(s)$ has an n-link (if the link is a k-link for some $k < n$ the following concepts are not relevant). The *link-ancestor* of v_m is the configuration v_j where the original copy of the n-link in $top_1(s)$ was created[8], or v_0 if the link was present in the stack of the configuration v_0. The *link-rank* of v_m is the minimum colour of a state occurring in Λ since its link-ancestor v_j (inclusive) *i.e.* it is $\min\{\Omega(v_j), \cdots \Omega(v_m)\}$.

Example 20. *Consider the sequence of indexed stacks given in Figure 4. The link-ancestor of configuration v'_8 is configuration v'_7 and its link-rank is* 5*. The link-ancestor of configuration v'_{11} is configuration v'_5 and its link-rank is* 2*.*

[8]Formally, one could index links as well: whenever performing, in configuration v_j, a $push_1^{\gamma,e}$, one attaches to the newly created link the index $j+1$. Later, if the link is copied (by doing a $push_k$ operation) then the index is copied as well.

Definition 8. *An n-CPDP $\mathcal{A} = \langle \Gamma, Q, \Delta, q_0 \rangle$ equipped with a colouring function is* rank-aware *from a configuration v_0 if there exist a function $LinkRk : Q \times \Gamma \to \mathbb{N}$ such that for any finite path $\Lambda = v_0 v_1 \cdots v_\ell$, the link-rank (if defined) of the configuration $v_\ell = (q, s)$ is equal to $LinkRk(q, top_1(s))$. In other words, the link rank can be retrieved from the control state together with the top_1-element of the stack.*

Remark 7. *In the current setting, if the ancestor of the pointed stack (*resp *the ancestor of the the top_1-element of $pop_k(s)$ / the link-ancestor) is v_0, then the collapse-rank (*resp *the pop-rank / the link-rank) is simply the smallest colour seen since the beginning of the play. Hence, it does not make much sense but it permits the construction to remain uniform.*

The next theorem shows that we can restrict our attention to CPDP games where the underlying CPDP is rank-aware.

Theorem 6. *For any n-CPDP $\mathcal{A} = \langle \Gamma, Q, \Delta, q_0 \rangle$ and any associated parity game $\mathbb{G}$, one can construct an n-CPDP $\mathcal{A}_{\rm rk}$ and an associated parity game $\mathbb{G}_{\rm rk}$ such that the following holds.*

- *There exists a mapping ν from the configurations of $\mathcal{A}$ to that of $\mathcal{A}_{\rm rk}$ such that:*
  - *for any configuration v_0 of $\mathcal{A}$, $\mathcal{A}_{\rm rk}$ is rank-aware from $\nu(v_0)$;*
  - *Éloïse has a winning strategy in $\mathbb{G}$ from some configuration v_0 iff she has a winning strategy in $\mathbb{G}_{\rm rk}$ from $\nu(v_0)$;*
- *If there is an n-CPDA transducer $\mathcal{S}_{\rm rk}$ synchronised with $\mathcal{A}_{\rm rk}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\rm rk}$ from $\nu(q_0, \bot_n)$, then one can effectively construct an n-CPDA transducer $\mathcal{S}$ synchronised with $\mathcal{A}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}$ from the initial configuration $(q_0, \bot_n)$.*

The proof of Theorem 6 is a non-trivial generalisation of [A4, Lemma 6.3] (which concerns 2-CPDP) to the general setting of n-CPDP and starting from an arbitrary configuration. The rest if the subsection is devoted to this proof.

Fix an n-CPDP $\mathcal{A} = \langle \Gamma, Q, \Delta, q_0 \rangle$, a partition $Q_{\mathbf{E}} \uplus Q_{\mathbf{A}}$ of Q and a colouring function $\Omega : Q \to C \subset \mathbb{N}$. Denote by $\mathbb{G}$ the induced parity game. We define a rank-aware (to be proven) n-CPDP $\mathcal{A}_{\rm rk} = \langle \Gamma_{\rm rk}, Q_{\rm rk}, \Delta_{\rm rk}, q_{0,\rm rk} \rangle$ such that $Q_{\rm rk} = Q \times C$ and

$$\Gamma_{\rm rk} = \Gamma \times (C \cup \{\circlearrowleft\}) \times (C \cup \{\circlearrowleft, \dagger\}) \times (C^{\{1,\dots,n\}} \cup \{\circlearrowleft\})$$

We define a map ν that associates with any configuration of $\mathcal{A}$ a configuration of $\mathcal{A}_{\rm rk}$. Let (q, s) be a configuration in $\mathcal{A}$. Then $\nu(q, s) = ((q, \Omega(q)), s')$ where s' is obtained by:

- Replacing every internal symbol γ (*i.e.* that is not the top_1-element) by $(\gamma, \circlearrowleft, \circlearrowleft, \circlearrowleft)$ if it has an n-link and by $(\gamma, \circlearrowleft, \dagger, \circlearrowleft)$ otherwise.
- Replacing the top_1-element γ by $(\gamma, \Omega(q), \Omega(q), \Omega(q))$ if it has an n-link and otherwise by $(\gamma, \Omega(q), \dagger, \Omega(q))$.

We equip $\mathcal{A}_{\rm rk}$ with a colouring function $\Omega_{\rm rk}$ by letting $\Omega_{\rm rk}(q, \theta) = \Omega(q)$. Our construction will satisfy the following invariant. Let Λ be a finite path (in $\mathrm{Graph}(\mathcal{A}_{\rm rk})$) starting in some configuration $\nu(q, s)$ ending in some configuration $((q, \theta), s)$ then the following holds. First, θ is the minimal colour visited from the beginning of the path. Second, if $top_1(s) = (\alpha, m_c, m_l, \tau)$ then

- m_c is the collapse-rank;
- m_l is the link-rank if it makes sense (*i.e.* there is an n-link in the current top_1-symbol) or is $\dagger$ otherwise;
- τ is the *pop-rank*: $\tau(i)$ is the pop-rank for i for every $1 \leqslant i \leqslant n$.

Let us now explain how ν is defined. Let (q, s) be some configuration in $\mathcal{A}$. Then $\nu(q, s) = ((q, \Omega(q), s')$ where s' is obtained by:

- Replacing every internal symbol γ (*i.e.* that is not the top_1-element) by $(\gamma, \circlearrowleft, \circlearrowleft, \circlearrowleft)$ if it has an n-link and by $(\gamma, \circlearrowleft, \dagger, \circlearrowleft)$ otherwise.
- Replacing the top_1-element γ by $(\gamma, \Omega(q), \Omega(q), \Omega(q))$ if it has an n-link and otherwise by $(\gamma, \Omega(q), \dagger, \Omega(q))$.

Trivially, at the beginning of the path the invariant holds.

The transition function of $\mathcal{A}_{\rm rk}$ mimics that of $\mathcal{A}$ and updates the ranks as explained below. First, let us explain the meaning of symbols $\circlearrowleft$. Such symbols will never been created using a $push_1^{\neg,k}$ or a $rew_1^{\circlearrowleft}$ action: hence they can only be duplicated (using $push_k$) from symbols originally in the stack. The meaning of a symbol $\circlearrowleft$ is that the corresponding object (collapse-rank, link-rank or pop-rank) has not yet been settled. However, when a $\circlearrowleft$ symbol appears in the top_1-element the various ranks can be easily retireved as they necessarily equal the smallest colour visited so far (as noted in Remark 7): this is why we made the computation of the minimal colour visited so far in the control state of $\mathcal{A}_{\rm rk}$.

In order to make the construction more readable, we do not formally describe $\Delta_{\rm rk}$ but rather explain how $\mathcal{A}_{\rm rk}$ behaves. It should be clear that $\Delta_{\rm rk}$ can be formally described to fit this informal description (and that some extra control states are actually needed as we allow to do several stack operation per transition); technical issues about this construction are

discussed in Remark 8. Note that the description below also contains the inductive proof of its validity, namely that m_c, m_l and τ are as stated above. To avoid case distinction on whether the link-rank is defined or not, we take the following convention that $\min(\dagger, i) = \dagger$ for every $i \in \mathbb{N}$.

The intuitive idea is the following. One stores in the stack information on the various ranks, and after performing a pop_k or a $collapse$, one needs to update the information stored in the new top_1-element. Indeed this information has no longer been updated since the ancestor configuration (this was the last time it was on top of the stack). To update it, one uses either the collapse-rank / pop-rank in the previous configuration, which is exactly what is needed for this update.

Assume $\mathcal{A}_{\mathrm{rk}}$ is in configuration $v_\ell = ((q,\theta),s)$ with $top_1(s) = (\alpha, m_c, m_l, \tau)$ and let $v_0 v_1 \cdots v_\ell$ be the beginning of the path of $\mathrm{Graph}(\mathcal{A}_{\mathrm{rk}})$ where we denote $v_i = ((q_i,\theta_i),s_i)$ (hence $q_\ell = q$ and $s_\ell = s$). For any $(q', rew_1^\gamma; op) \in \Delta(q,\alpha)$ (note that the case where no rew_1 is performed corresponds to the case where $\gamma = \alpha$) the following behaviours are those allowed in $((q,\theta),s)$.

1) Assume $op = pop_k$ for some $1 \leqslant k \leqslant n$, let $pop_k(s) = s'$ and let $top_1(s') = (\alpha', m'_c, m'_l, \tau')$. Then $\mathcal{A}_{\mathrm{rk}}$ can go to the configuration $((q',\theta'),s'')$ where $\theta' = \min(\theta, \Omega(q'))$ and s'' is obtained from s' by replacing $top_1(s')$ by
   a) $(\alpha', \theta', \theta', (\theta', \dots, \theta'))$ if $m'_c = \circlearrowleft$, $m'_l = \circlearrowleft$ and $\tau' = (\circlearrowleft, \dots, \circlearrowleft)$.
   b) $(\alpha', \theta', \dagger, (\theta', \dots, \theta'))$ if $m'_c = \circlearrowleft$, $m'_l = \dagger$ and $\tau' = (\circlearrowleft, \dots, \circlearrowleft)$.
   c) $(\alpha', \min(m'_c, \tau(k), \Omega(q')), \min(m'_l, \tau(k), \Omega(q')), \tau'')$, with

$$\tau''(i) = \begin{cases} \min(\tau'(i), \tau(k), \Omega(q')) & \text{if } i \leqslant k \\ \min(\tau(i), \Omega(q')) & \text{if } i > k \end{cases}$$

   Cases (a) and (b) correspond to the case where one reach (possibly a copy) of a symbol that was in the stack from the very beginning and that never appeared as a top_1-element: then the value of the collapse-rank, link-rank (if defined this is case (a) otherwise it is case (b)) and pop-ranks are all equal to θ'.
   We now explain case (c). Let v_x be the ancestor of $top_1(pop_k(s))$. Then $x > 0$ as otherwise we would be in case (a) or (b). By Proposition 4, it follows that $top_1(pop_k(s)) = top_1(s_{x-1})$, and by induction hypothesis, at step $(x-1)$, m'_c, m'_l and τ' had the expected meaning. Let y be the index of the top_1-element of the pointed stack in s': y is also the top_1-element of the pointed stack in s_{x-1}, and moreover $y < x$. The collapse-rank in $v_{\ell+1}$ is

$$\begin{aligned} &\min\{\Omega(q_y), \dots, \Omega(q_{x-1}), \Omega(q_x), \dots, \Omega(q_n), \Omega(q')\} \\ &= \min\{\min\{\Omega(q_y), \dots, \Omega(q_{x-1})\}, \min\{\Omega(q_x), \dots, \Omega(q_n)\}, \Omega(q')\} \\ &= \min\{m'_c, \tau(k), \Omega(q')\} \end{aligned}$$

   Similarly, when defined, the link-ancestor of s' is the same as the one in s_{x-1}: hence the pop-rank in $v_{\ell+1}$ is $\min\{m'_l, \tau(k), \Omega(q')\}$.
   For any $i \leqslant k$, $top_1(pop_i(s')) = top_1(s_{x-1})$ and therefore the pop-rank for i in $v_{\ell+1}$ is obtained by updating $\tau'(i)$ to take care of the minimum colour seen since v_x – which (as for the collapse-rank) is $\min\{\tau(k), \Omega(q')\}$: therefore the pop-rank for i in $v_{\ell+1}$ equals $\min\{\tau'(i), \tau(k), \Omega(q')\}$.
   For any $i > k$, $pop_i(s') = pop_i(s)$ and thus $top_1(pop_i(s')) = top_1(pop_i(s))$. Therefore the pop-rank for i in $v_{\ell+1}$ is obtained by updating the one in v_ℓ to take care of the new visited colour $\Omega(q')$: hence the pop-rank for i in $v_{\ell+1}$ equals $\min\{\tau(i), \Omega(q')\}$.

2) Assume $op = collapse$, let $collapse(s) = s'$ and let $top_1(s') = (\alpha', m'_c, m'_l, \tau')$. Then $\mathcal{A}_{\mathrm{rk}}$ can go to the configuration $((q',\theta'),s'')$ where $\theta' = \min(\theta, \Omega(q'))$ and s'' is obtained from s' by replacing $top_1(s')$ by
   a) $(\alpha', \theta', \theta', (\theta', \dots, \theta'))$ if $m'_c = \circlearrowleft$, $m'_l = \circlearrowleft$ and $\tau' = (\circlearrowleft, \dots, \circlearrowleft)$.
   b) $(\alpha', \theta', \dagger, (\theta', \dots, \theta'))$ if $m'_c = \circlearrowleft$, $m'_l = \dagger$ and $\tau' = (\circlearrowleft, \dots, \circlearrowleft)$.
   c) $(\alpha', \min(m'_c, m_c, \Omega(q')), \min(m'_l, m_c, \Omega(q')), \tau'')$ with

$$\tau''(i) = \begin{cases} \min(\tau'(i), m_c, \Omega(q')) & \text{if } i \leqslant k \\ \min(\tau(i), \Omega(q')) & \text{if } i > k \end{cases}$$

   The proof follows the same line as for the previous case (pop_k).
   Cases (a) and (b) correspond to the case where one reach (possibly a copy) of a symbol that was in the stack from the very beginning and that never appeared as a top_1-element: then the value of the collapse-rank, link-rank (if defined this is case (a) otherwise it is case (b)) and pop-ranks are all equal to θ'.
   We now explain case (c). Let v_x be the collapse-ancestor of v_ℓ. Then $x > 0$ as otherwise we would be in case (a) or (b). By induction hypothesis, m'_c and τ' give the collapse-rank / link-rank / pop-ranks in v_{x-1}. Moreover the ancestor

of the top_1-element of the target of the top link in s' is the same as the one in v_{x-1}. Therefore the collapse-rank is obtained by taking the minimum of the collapse-rank in v_{x-1} with $\min\{\Omega(q_x), \dots \Omega(q_n), \Omega(q')\} = \min\{m_c, \Omega(q')\}$. Similarly (if defined) the link-ancestor in s' being the same as the one in v_{x-1}, the link-rank is obtained by taking the minimum of the one in v_{x-1} with $\min\{\Omega(q_x), \dots \Omega(q_n), \Omega(q')\} = \min\{m_c, \Omega(q')\}$.
Let $i \leqslant k$. The ancestor of $top_1(pop_i(s'))$ is the same as the ancestor of $top_1(pop_i(s_{x-1}))$. Therefore the pop-rank for i in $v_{\ell+1}$ is obtained by taking the minimum of the one in v_{x-1} with $\min\{\Omega(q_x), \dots \Omega(q_n), \Omega(q')\} = \min\{m_c, \Omega(q')\}$.
Let $i > k$. Then the ancestor of $top_1(pop_i(s'))$ is the same as the ancestor of $top_1(pop_i(s_n))$: indeed the collapse only modified the top_k stack. Therefore the pop-rank for i in $v_{\ell+1}$ is obtained by taking the minimum of the one in v_ℓ with the new visited colour $\Omega(q')$.

3) Assume $op = push_j$ for some $2 \leqslant j \leqslant n$, let $push_j(rew_1^{(\gamma, m_c, m_l, \tau)}(s)) = s'$ and let $top_1(s') = (\gamma, m_c, m_l, \tau)$ (note that $\circlearrowleft$ does not appear in $top_1(s')$). Then, $\mathcal{A}_{\mathrm{rk}}$ can go to the configuration $((q', \theta'), s'')$ where $\theta' = \min(\theta, \Omega(q'))$ and s'' is obtained from s' when replacing $top_1(s')$ by $(\gamma, \min(m_c, \Omega(q')), \min(m_l, \Omega(q')), \tau')$ with

$$\tau'(i) = \begin{cases} \min(\tau(i), \Omega(q')) & \text{if } i \neq j \\ \Omega(q') & \text{if } i = j \end{cases}$$

Indeed, the collapse-ancestor in the new configuration is the same as the one in s. As by induction hypothesis m_c is the collapse-rank in v_ℓ, the collapse-rank in $v_{\ell+1}$ is obtained by updating m_c to take care of the new visited colour, namely by taking $\min\{m_c, \Omega(q')\}$. Similarly, if defined, the link-ancestors in v_ℓ and $v_{\ell+1}$ are identical and then the link-rank in $v_{\ell+1}$ is $\min\{m_c, \Omega(q')\}$.
For any $i \neq j$, the ancestor of $top_1(pop_i(s)')$ and the ancestor of $top_1(pop_i(s'))$ are the same. Again using the induction hypothesis one directly gets that the pop-rank for i in $v_{\ell+1}$ equals $\min\{\tau(i), \Omega(q')\}$.
The index of the ancestor of $top_1(pop_j(s'))$ is by definition $\ell + 1$. Hence as the only colour visited since $v_{\ell+1}$ is $\Omega(q')$ it equals the pop-rank for j.

4) Assume $op = push_1^{\beta,k}$ with $1 \leqslant k \leqslant n$, and $\beta \in (\Gamma \backslash \{\bot\})$. Then $\mathcal{A}_{\mathrm{rk}}$ can go to (q', θ'), where $\theta' = \min(\theta, \Omega'(q'))$, and apply successively $rew_1^{(\gamma, m_c, m_l, \tau)}$ and $push_1^{(\beta, m_c', m_l', \tau'), k}$ where $m_c' = \min(\tau(k), \Omega(q'))$, $m_l' = \Omega(q')$ if $k = n$ and $m_l' = \dagger$ otherwise, and $\tau'(i) = \min(\tau(i), \Omega(q'))$ for every $i \geqslant 2$ and $\tau(1) = \Omega(q')$.
Indeed, the pointed stack in s' is $top_k(pop_k(s))$ and therefore the collapse-rank in $v_{\ell+1}$ is the minimum of the pop-rank for k in s and of the new visited colour $\Omega(q')$, that is $\min\{\tau(k), \Omega(q')\}$.
If $k = n$, the link-ancestor of $v_{\ell+1}$ is $v_{\ell+1}$ itself and hence the link-rank is the colour of the current configuration, namely $\Omega(q')$.
For any $i \geqslant 2$, as $pop_i(s) = pop_i(s')$ one also has $top_1(pop_i(s')) = top_1(pop_i(s))$ and therefore the pop-rank for i in $v_{\ell+1}$ equals the minimum of the one in v_ℓ with the new visited colour $\Omega(q')$, that is $\min\{\tau(i), \Omega(q')\}$. Finally as the ancestor of $pop_1(s')$ is $v_{\ell+1}$ then the pop-rank for 1 is the current colour, namely $\Omega(q')$.

From the previous description (and the included inductive proof) we conclude that, for any configuration v_0 of $\mathcal{A}$, $\mathcal{A}_{\mathrm{rk}}$ is rank-aware from $\nu(v_0)$.

Remark 8. *One must object that $\mathcal{A}_{\mathrm{rk}}$ does not fit the definition of n-CPDP. Indeed, in a single transition it can do a top-rewriting followed by another stack operation and followed again by a top-rewriting (which itself depends on the new top_1-element). One could add intermediate states and simply decompose such a transition into two transitions, but this would be problematic later when defining an n-CPDA transducer realising a winning strategy.*

Hopefully, one can define a variant $\mathcal{A}'_{\mathrm{rk}}$ of $\mathcal{A}_{\mathrm{rk}}$ that has the same properties as $\mathcal{A}_{\mathrm{rk}}$ and additionally fits the definition of n-CPDP. The idea is simply to postpone the final top-rewriting to the next transition. Indeed, it suffices to add a new component on the control state where one encodes the top-rewriting that should be performed next: this top-rewriting is then performed in the next transition (note that this fits the definition as performing two top-rewriting is the same as only doing the last one). However, there is still an issue as the top-rewriting was actually depending on the top_1-symbol (one updates the various ranks) hence, one cannot save the next top-rewriting in the control state without first observing the symbol to be rewritten. But this is not a problem, as it suffices to remember which kind of update should be done (one concerning a pop_k or one concerning a $collapse$) and to store in the control state the various objects needed for this update (for this one can simply store the former top_1-element).

One also need to slightly modify the $LinkRk$ function so that it return the link-rank of the top_1-symbol after it is rewritten. This can easily be done as the domain of $LinkRk$ is $Q_{\mathrm{rk}} \times \Gamma_{\mathrm{rk}}$.

Note that $\mathcal{A}'_{\mathrm{rk}}$ and $\mathcal{A}_{\mathrm{rk}}$ use the same stack alphabet, but that the state space of $\mathcal{A}_{\mathrm{rk}}$ uses an extra component of size linear in the one of the stack alphabet.

In conclusion building a rank-aware (valid) n-CPDP from a non-aware one increases (by a multiplicative factor) the stack alphabet by C^{n+3} and the state set by $\mathcal{O}(C^{n+3})$.

For now on, we uses $\mathcal{A}_{\mathrm{rk}}$ to mean $\mathcal{A}'_{\mathrm{rk}}$.

We are now ready to conclude the proof of Theorem 6. First recall that we defined Ω_{rk} by letting $\Omega_{\mathrm{rk}}(q,\theta) = \Omega(q)$. Then, we define a partition $Q_{\mathrm{rk},\mathbf{E}} \uplus Q_{\mathrm{rk},\mathbf{A}}$ of Q_{rk} by letting the states in $Q_{\mathrm{rk},\mathbf{E}}$ to be those states with their first component in $Q_{\mathbf{E}}$, and those states in $Q_{\mathrm{rk},\mathbf{A}}$ to be those states with their first component in $Q_{\mathbf{A}}$. Let $\mathcal{G}_{\mathrm{rk}}$ be the corresponding arena and let $\mathbb{G}_{\mathrm{rk}}$ be the corresponding n-CPDP parity game.

Consider the projection ζ defined from configurations of $\mathcal{A}_{\mathrm{rk}}$ into configurations of $\mathcal{A}$ by only keeping the first component of the control state, and by only keeping the Γ part of symbols appearing in the stack. Note that, on the domain of ν^{-1}, ζ and ν^{-1} coincide. Also note that ζ preserve the shape of stacks[9], *i.e.* for any configuration v_{rk}, the stack in v_{rk} has the same shape as the stack in $\nu(v_{\mathrm{rk}})$.

We extend ζ as a function from (possibly partial) plays in $\mathbb{G}_{\mathrm{rk}}$ into (possibly partial) plays in $\mathbb{G}$ by letting $\zeta(v'_0 v'_1 \cdots) = \zeta(v'_0)\zeta(v'_1)\cdots$. It is obvious that for any play Λ' in $\mathbb{G}_{\mathrm{rk}}$ starting from $\nu(v_0)$, its image $\zeta(\Lambda')$ is a play in $\mathbb{G}$ starting from v_0; moreover these two plays induces the same sequence of colours and at any round the player that controls the current configuration is the same in both plays. Conversely, from the definition of $\mathcal{A}_{\mathrm{rk}}$ it is also clear that there is, for any play Λ in $\mathbb{G}$ starting from v_0, a *unique* play Λ' in $\mathbb{G}_{\mathrm{rk}}$ starting from $\nu(v_0)$ such that $\zeta(\Lambda') = \Lambda$.

In particular, ζ can be used to construct a strategy in $\mathbb{G}$ from a strategy in $\mathbb{G}_{\mathrm{rk}}$. Indeed, let Φ_{rk} be a strategy for Éloïse from $\nu(v_0)$ in $\mathbb{G}_{\mathrm{rk}}$. We define a strategy Φ in $\mathbb{G}$ from $\nu(v_0)$. This strategy maintains as a memory a partial play Λ_{rk} in $\mathbb{G}_{\mathrm{rk}}$ such that, if Éloïse respects Φ, in $\mathbb{G}$ starting from v_0 after having played Λ one has $\zeta(\Lambda_{\mathrm{rk}}) = \Lambda$ and moreover Λ_{rk} is a play in $\mathbb{G}_{\mathrm{rk}}$ starting from $\nu(v_0)$ where Éloïse respects Φ_{rk}. Initially, we let $\Lambda_{\mathrm{rk}} = \nu(v_0)$. Assume that we have been playing Λ and that Éloïse has to play next. Then she considers $v_{\mathrm{rk}} = \Phi_{\mathrm{rk}}(\Lambda_{\mathrm{rk}})$ and she plays to v where v is the unique configuration such that $\zeta(\Lambda_{\mathrm{rk}} \cdot v_{\mathrm{rk}}) = \Lambda \cdot v$. Finally one updates Λ_{rk} to be $\Lambda_{\mathrm{rk}} \cdot v_{\mathrm{rk}}$. If it is Abelard that has to play next and if he moves to some v, then Éloïse updates Λ_{rk} to be $\Lambda_{\mathrm{rk}} \cdot v_{\mathrm{rk}}$ where v_{rk} is the unique configuration such that $\Lambda_{\mathrm{rk}} \cdot v_{\mathrm{rk}}$ is a valid play and such that $\zeta(v_{\mathrm{rk}}) = v$. A symmetrical construction can be done to build a strategy of Abelard in $\mathbb{G}$ from one in $\mathbb{G}_{\mathrm{rk}}$.

Now, assume that $\nu(v_0)$ is winning for Éloïse (*resp.* Abelard) and call Φ_{rk} an associated winning strategy. Let Φ be the strategy in $\mathbb{G}$ obtained as explained above. Then Φ is winning for Éloïse (*resp.* Abelard) in $\mathbb{G}$ from v_0 (this follows directly from the fact that Φ_{rk} is winning and that we have the property that $\zeta(\Lambda_{\mathrm{rk}}) = \Lambda$ for any partial play Λ in $\mathbb{G}$ consistent with Φ). Hence this proves that Éloïse has a winning strategy in $\mathbb{G}$ from v_0 iff she has a winning strategy in $\mathbb{G}_{\mathrm{rk}}$ from $\nu(v_0)$.

Finally, from the previous construction of a strategy Φ from a strategy Φ_{rk} we prove that if there is an n-CPDA transducer $\mathcal{S}_{\mathrm{rk}}$ synchronised with $\mathcal{A}_{\mathrm{rk}}$ realising a well-defined winning strategy Φ_{rk} for Éloïse in $\mathbb{G}_{\mathrm{rk}}$ from $\nu(q_0, \bot_n)$, then one can effectively construct an n-CPDA transducer $\mathcal{S}$ synchronised with $\mathcal{A}$ realising a well-defined winning strategy Φ for Éloïse in $\mathbb{G}$ from the initial configuration $(q_0, \bot_n)$. Indeed, in our previous construction of Φ, we maintained a partial play Λ_{rk} in $\mathbb{G}_{\mathrm{rk}}$ and used the value of $\Phi_{\mathrm{rk}}(\Lambda_{\mathrm{rk}})$ to define $\Phi(\Lambda)$. But if Φ_{rk} is realised by an n-CPDA transducer $\mathcal{S}_{\mathrm{rk}}$, it suffices to remember the configuration of this transducer after playing Λ_{rk} (as this suffices to compute $\varphi_{\mathrm{rk}}(\Lambda_{\mathrm{rk}})$). Hence, the only things that need to be modified from $\mathcal{S}_{\mathrm{rk}}$ to obtain $\mathcal{S}$ is that one needs to "embed" the transition function of $\mathcal{A}_{\mathrm{rk}}$ into it, so that $\mathcal{S}$ can read/output elements in $Q \times Op_n(\Gamma) \times Op_n(\Gamma)$ instead of $Q_{\mathrm{rk}} \times Op_n(\Gamma_{\mathrm{rk}}) \times Op_n(\Gamma_{\mathrm{rk}})$. This can easily (but writing the formal construction would be quite heavy) be achieved by noting that the shape of stacks is preserved by ζ: hence if $\mathcal{S}_{\mathrm{rk}}$ is synchronised with $\mathcal{A}_{\mathrm{rk}}$ then $\mathcal{S}$ is synchronised with $\mathcal{A}$ (as $\mathcal{A}_{\mathrm{rk}}$ and $\mathcal{A}$ are "synchronised", and $\mathcal{S}_{\mathrm{rk}}$ and $\mathcal{S}$ are "synchronised" as well).

If we summarise, the overall blowup in the transformation from $\mathbb{G}$ to $\mathbb{G}_{\mathrm{rk}}$ given by Theorem 6 is as follows.

Proposition 5. *Let $\mathcal{A}$ and $\mathcal{A}_{\mathrm{rk}}$ be as in Theorem 6. Let $\{0, \cdots, d\}$ be the set of colours. Then the set of states of $\mathcal{A}_{\mathrm{rk}}$ has size $\mathcal{O}(|Q|(d+1)^{n+3})$ and the stack alphabet of $\mathcal{A}_{\mathrm{rk}}$ has size $\mathcal{O}(|\Gamma|(d+1)^{2n+5})$.*

Moreover the set of colours used in $\mathbb{G}$ and $\mathbb{G}_{\mathrm{rk}}$ are the same.

Proof: By construction together with Remark 8. ∎

6) Removing the n-links:

In this subsection, we show how one can remove the outmost links.

Theorem 7. *For any* rank-aware *n-CPDP $\mathcal{A}_{\mathrm{rk}} = \langle \Gamma_{\mathrm{rk}}, Q_{\mathrm{rk}}, \Delta_{\mathrm{rk}}, q_{0,\mathrm{rk}} \rangle$ and any associated parity game $\mathbb{G}_{\mathrm{rk}}$, one can construct an n-CPDP $\mathcal{A}_{\mathrm{lf}}$ and an associated parity game $\mathbb{G}_{\mathrm{lf}}$ such that the following holds.*

- *$\mathcal{A}_{\mathrm{lf}}$ does not create n-links.*

[9]Recall that the *shape* of a stack is the stack obtained by replacing all symbols appearing in s by a fresh symbol $\sharp$ (but keeping the links).

- *There exists a mapping ν from the configurations of $\mathcal{A}_{\mathrm{rk}}$ to that of $\mathcal{A}_{\mathrm{lf}}$ such that:*
  - *Éloïse has a winning strategy in $\mathbb{G}_{\mathrm{rk}}$ from some configuration v_0 iff she has a winning strategy in $\mathbb{G}_{\mathrm{lf}}$ from $\nu(v_0)$;*
- *If there is an n-CPDA transducer $\mathcal{S}_{\mathrm{lf}}$ synchronised with $\mathcal{A}_{\mathrm{lf}}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ from $\nu(q_{0,\mathrm{rk}}, \perp_n)$, then one can effectively construct an n-CPDA transducer $\mathcal{S}_{\mathrm{rk}}$ synchronised with $\mathcal{A}_{\mathrm{rk}}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\mathrm{rk}}$ from the initial configuration $(q_{0,\mathrm{rk}}, \perp_n)$.*

The whole section is devoted to the proof of Theorem 7 and we thus fix from now on, a *rank-aware* n-CPDP $\mathcal{A}_{\mathrm{rk}} = \langle \Gamma_{\mathrm{rk}}, Q_{\mathrm{rk}}, \Delta_{\mathrm{rk}}, q_{0,\mathrm{rk}} \rangle$ (together with a function $LinkRk$), a partition $Q_{\mathrm{rk},\mathbf{E}} \uplus Q_{\mathrm{rk},\mathbf{A}}$ of Q_{rk}, a colouring function $\Omega : Q_{\mathrm{rk}} \to C \subset \mathbb{N}$ and we let $C = \{0, \dots, d\}$. Denote by G_{rk} the transition graph of $\mathcal{A}_{\mathrm{rk}}$, by $\mathcal{G}_{\mathrm{rk}}$ the arena induced by G_{rk} and the partition $Q_{\mathrm{rk},\mathbf{E}} \uplus Q_{\mathrm{rk},\mathbf{A}}$, and by $\mathbb{G}_{\mathrm{rk}}$ the parity game $(\mathcal{G}_{\mathrm{rk}}, \Omega)$.

Consider the following informal description of a new game $\mathbb{G}_{\mathrm{lf}}$ (here lf intend to mean *link-free*) defined from $\mathbb{G}_{\mathrm{rk}}$. The new game mimics $\mathbb{G}_{\mathrm{rk}}$ except that whenever a player wants to perform a $push_1^{\gamma,n}$ operation, this is replaced by the following "negotiation" between the players:

- First, Éloïse has to provide a vector $\overrightarrow{R} = (R_0, \cdots R_d) \in (2^{Q_{\mathrm{rk}}})^{d+1}$ whose intended meaning is the following: she claims that she has a strategy such that if the newly created n-link (or a copy of it) is eventually used by doing a $collapse$ then it leads to a state in R_i where i is the smallest colour visited since the original copy of the link was created.
- Then, Abelard has two options. He can agree with Éloïse's claim, pick a state q in some R_i and perform a pop_n action whilst going to state q (through an intermediate dummy vertex coloured by i): this is the case where Abelard wants to simulate a $collapse$ involving the n-link. Alternatively Abelard can decide to push the symbol $(\gamma, \overrightarrow{R})$ (and a dummy 1-link is attached).

Later in some configuration (q, s) with a top_1-element of the form $(\gamma, \overrightarrow{R})$ if the player controlling q wants to simulate a transition $(q', op; collapse)$ that collapses the stack, then this move is replaced by one that goes to a sink configuration that is winning for Éloïse iff $q' \in R_i$ where $i = LinkRk(q, \gamma)$ is the *link rank* and hence corresponds to the smallest colour visited since the original copy of symbol $(\gamma, \overrightarrow{R})$ was pushed onto the stack (recall that $\mathcal{A}_{\mathrm{rk}}$ is rank-aware). The intuitive idea is that, when simulating a $collapse$ (involving an order-n link), Éloïse wins iff her initial claim on the possible states reachable by following the link was correct. Otherwise she loses.

We now define $\mathcal{A}_{\mathrm{lf}}$ and the associated game $\mathbb{G}_{\mathrm{lf}}$. We start with an informal description of $\mathcal{A}_{\mathrm{lf}}$ and then formally describe its structure.

The n-CPDP $\mathcal{A}_{\mathrm{lf}}$ *simulates* $\mathcal{A}_{\mathrm{rk}}$ as follows. Assume that the play is in some configuration (q, s) and that the player that controls it wants to simulate a transition $(q', rew_1^{\alpha}; op) \in \Delta_{\mathrm{rk}}(q, top_1(s))$. In case op is neither of the form $push_1^{\beta,n}$ nor of the form $collapse$ with $top_1(s)$ having an n-link then the same transition $(q', rew_1^{\alpha}; op)$ is available in $\mathcal{A}_{\mathrm{rk}}$ and is performed. The interesting case is when $op = push_1^{\beta,n}$, and it is simulated by $\mathcal{A}_{\mathrm{lf}}$ as follows.

- The control state of $\mathcal{A}_{\mathrm{lf}}$ is updated to be q^{β} and one performs rew_1^{α}.
- From q^{β}, Éloïse has to move to a new control state $q^{?}$ and can push any symbol of the form $(\alpha, \overrightarrow{R})$ where $\overrightarrow{R} = (R_0, \cdots R_d) \in (2^Q)^{d+1}$. A dummy 1-link is attached (and will never be used for a $collapse$).
- From $q^{?}$, Abelard has to play and choose between one of the following two options:
  - either go to state q and perform no action on the stack,
  - or pick a state p in some R_i, go to an intermediate new state p^i (of colour i) without changing the stack and from this new configuration go to state p and perform a pop_n action.

The intended meaning of such a decomposition of the $push_1^{\beta,n}$ operation is the following: when choosing the sets in $\overrightarrow{R}$, Éloïse is claiming that she has a strategy such that if the n-link created by pushing β is eventually used for collapsing the stack then the control state after collapsing will belong to R_i where i is meant to be the smallest colour from the creation of the link to the collapse of the stack (equivalently it will be the link rank — as computed in $\mathcal{A}_{\mathrm{rk}}$ — the just before collapsing). Note that the R_i are arbitrary sets because Éloïse has not a full control on the play (and in general cannot force R_i to be a singleton). Then Abelard is offered to simulate the $collapse$ (here state p^i is only used for going through a state of colour i). If he does not want to simulate a $collapse$ then one stores $\overrightarrow{R}$ for possibly checking its truth later in the play.

Assume that later, in configuration (p', s') one of the two players wants to simulate a transition $(p'', rew_1^{\beta}; collapse)$ involving an n-link. By construction, $top_1(s')$ is necessarily of the form $(\gamma, \overrightarrow{R})$. Then the simulation is done by going to a sink configuration that is winning for Éloïse iff $p'' \in R_{LinkRk(p',\gamma)}$, *i.e.* Éloïse wins iff her former claim on $\overrightarrow{R}$ was correct.

Formally we set $\mathcal{A}_{\mathrm{lf}} = \langle \Gamma_{\mathrm{lf}}, Q_{\mathrm{lf}}, \Delta_{\mathrm{lf}}, q_{0,\mathrm{lf}} \rangle$ with

- $\Gamma_{\mathrm{lf}} = \Gamma_{\mathrm{rk}} \cup \Gamma_{\mathrm{rk}} \times (2^{Q_{\mathrm{rk}}})^{d+1}$
- $Q_{\mathrm{lf}} = Q_{\mathrm{rk}} \cup \{q^{\gamma} \mid q \in Q_{\mathrm{rk}},\ \gamma \in \Gamma_{\mathrm{rk}}\} \cup \{q^{?} \mid q \in Q_{\mathrm{rk}}\} \cup \{q^i \mid q \in Q_{\mathrm{rk}},\ 0 \leqslant i \leqslant d\} \cup \{tt, ff\}$

- Δ_{lf} is defined as follows, where q, q' range over Q_{rk}, α, β, γ range over Γ_{rk} and $\overrightarrow{R} = (R_0, \dots, R_d)$ ranges over $(2^{Q_{\mathrm{rk}}})^{d+1}$.
  - If $(q', rew_1^\alpha; op) \in \Delta_{\mathrm{rk}}(q, \gamma)$ and if op is neither of the form $push_1^{\beta,n}$ nor $collapse$, then $(q', rew_1^\alpha; op) \in \Delta_{\mathrm{lf}}(q, \gamma)$ and $(q', rew_1^{(\alpha, \overrightarrow{R})}; op) \in \Delta_{\mathrm{lf}}(q, (\gamma, \overrightarrow{R}))$.
  - If $(q', rew_1^\alpha; push_1^{\beta,n}) \in \Delta_{\mathrm{rk}}(q, \gamma)$, then $(q^\beta, rew_1^\alpha; id) \in \Delta_{\mathrm{lf}}(q, \gamma)$ and $(q^\beta, rew_1^{(\alpha, \overrightarrow{R})}; id) \in \Delta_{\mathrm{lf}}(q, (\gamma, \overrightarrow{R}))$.
  - For all $q^\beta \in Q_{\mathrm{lf}}$, $\Delta(q^\beta, \gamma) = \Delta(q^\beta, (\gamma, \overrightarrow{R})) = \{(q^?, push_1^{(\beta, \overrightarrow{S}),1}) \mid \overrightarrow{S} \in (2^{Q_{\mathrm{rk}}})^{d+1})\}$.
  - For all $q^? \in Q_{\mathrm{lf}}$, $\Delta(q^?, (\gamma, \overrightarrow{R})) = \{(q, id)\} \cup \{(p^i, id) \mid 0 \leqslant i \leqslant d \text{ and } p \in R_i\}$.
  - For all $q^i \in Q_{\mathrm{lf}}$, $\Delta(q^i, (\gamma, \overrightarrow{R})) = \{(q, pop_n)\}$.
  - If $(q', rew_1^\alpha; collapse) \in \Delta_{\mathrm{rk}}(q, \gamma)$, then $(q', rew_1^\alpha; collapse) \in \Delta_{\mathrm{lf}}(q, \gamma)$.
  - If $(q', rew_1^\alpha; collapse) \in \Delta_{\mathrm{rk}}(q, \gamma)$, then $(\lightning, id) \in \Delta_{\mathrm{lf}}(q, (\gamma, \overrightarrow{R}))$ if $q' \in R_{LinkRk(q,\gamma)}$ and $(\lightning\!\!\lightning, id) \in \Delta_{\mathrm{lf}}(q, (\gamma, \overrightarrow{R}))$ if $q' \notin R_{LinkRk(q,\gamma)}$.
  - $\Delta_{\mathrm{lf}}(\lightning, (\gamma, \overrightarrow{R})) = \{(\lightning, id)\}$ and $\Delta_{\mathrm{lf}}(\lightning\!\!\lightning, (\gamma, \overrightarrow{R})) = \{(\lightning\!\!\lightning, id)\}$.

We let G_{lf} be the transition graph of $\mathcal{A}_{\mathrm{lf}}$. Now, in order to define a game graph $\mathcal{G}_{\mathrm{lf}}$ out of G_{lf} we let $Q_{\mathrm{lf},\mathbf{E}} = Q_{\mathrm{rk},\mathbf{E}} \cup \{q^\gamma \mid q \in Q_{\mathrm{rk}},\ \gamma \in \Gamma_{\mathrm{rk}}\}$. Finally to define a corresponding n-CPDP parity game $\mathbb{G}_{\mathrm{lf}}$ we extend Ω by letting, $\forall q \in Q_{\mathrm{rk}}$ and $\gamma \in \Gamma_{\mathrm{rk}}$, $\Omega(q^\gamma) = \Omega(q^?) = d$ (as one cannot loop forever in such states, it means that they have no influence on the parity condition), $\Omega(q^i) = i$ for every $0 \leqslant i \leqslant d$, $\Omega(\lightning) = 0$ and $\Omega(\lightning\!\!\lightning) = 1$ (hence a play that visits $\lightning$ is winning for Éloïse and a play that visits $\lightning\!\!\lightning$ is winning for Abelard, as these states are sinks).

Note that $\mathcal{A}_{\mathrm{lf}}$ never create an n-link.

Consider some configuration $v_0 = (p_0, s_0)$ in $\mathbb{G}_{\mathrm{rk}}$. We explain now how to define an "equivalent" configuration $\nu(v_0)$ in $\mathbb{G}_{\mathrm{lf}}$ (here equivalent is in the sense of Theorem 8). The transformation consists in replacing any occurrence of a stack letter (call it γ) with an n-link in s_0 by another letter of the form $(\gamma, \overrightarrow{R})$ and replace the n-link by a 1-link. The vector $\overrightarrow{R}$ is defined as follows. Let s' be the stack obtained by popping every element and stack above γ, and let $R = \{q \mid$ Éloïse wins in $\mathbb{G}_{\mathrm{rk}}$ from $(q, collapse(s'))\}$. Then one sets $\overrightarrow{R} = (R, \cdots, R)$.

Example 21. *Assume we are playing a two-colour parity game and let*

$$s_0 = [\,[\,[\,a\,]\,]\,[\,[\,]\,[\,a\,b\,c\,]\,]\,[\,[\,]\,[\,a\,b\,c\,d\,]\,]\,],$$

$$R_1 = \{r \mid (r, [\,[\,[a]\,]\,]) \textit{ is winning for Éloïse in } \mathbb{G}_{\mathrm{rk}}\}$$

$$R_2 = \{r \mid (r, [\,[\,[\,a\,]\,]\,[\,[\,]\,[\,a\,b\,c\,]\,]\,]) \textit{ is winning for Éloïse in } \mathbb{G}_{\mathrm{rk}}\}$$

Then

$$\nu(s_0) = [\,[\,[\,a\,]\,]\,[\,[\,]\,[\,a\,b\,(c,(R_1,R_1))\,]\,]\,[\,[\,]\,[\,a\,b\,(c,(R_1,R_1))\,(d,(R_2,R_2))\,]\,]\,].$$

The rest of this section is devoted to the proof of the following result.

Theorem 8. *Éloïse wins in $\mathbb{G}_{\mathrm{rk}}$ from some configuration v_0 if and only if she wins in $\mathbb{G}_{\mathrm{lf}}$ from $\nu(v_0)$.*

Assume that the configuration $v_0 = (p_0, s_0)$ is winning for Éloïse in $\mathbb{G}_{\mathrm{rk}}$, and let Φ_{rk} be a winning strategy for her. Using Φ_{rk}, we define a strategy Φ_{lf} for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ from $\nu(v_0)$. The strategy Φ_{lf} maintains as a memory a partial play Λ_{rk} in $\mathbb{G}_{\mathrm{rk}}$, that is an element in V_{rk}^* (where V_{rk} denotes the set of vertices of G_{rk}). At the beginning Λ_{rk} is initialised to be (p_0, s_0). The play Λ_{rk} will satisfy the following invariant: assume that the play ends in a configuration (p, s), then the last configuration in Λ_{rk} has control state p and its top_1-element is either $top_1(s)$ or $(top_1(s), \overrightarrow{R})$ for some $\overrightarrow{R}$ (and in this case there is an n-link from the top_1-symbol of s).

We first describe Φ_{lf}, and then we explain how Λ_{rk} is updated.

Choice of the move. Assume that the play is in some vertex (p, s) with $p \in Q_{\mathrm{lf},\mathbf{E}} \backslash \{q^\gamma \mid q \in Q_{\mathrm{rk}},\ \gamma \in \Gamma_{\mathrm{rk}}\}$. The move given by Φ_{lf} depends on $\Phi_{\mathrm{rk}}(\Lambda_{\mathrm{rk}}) = (q, rew_1^\alpha; op)$ (we shall later argue that Φ_{lf} is well defined while proving that it is winning).

- If op is neither of the form $push_1^{\beta,n}$ nor $collapse$ then Éloïse plays $(q, rew_1^\alpha; op)$ if $top_1(s) = \gamma$ and she plays $(q, rew_1^{(\alpha, \overrightarrow{R})}; op)$ if $top_1(s) = (\gamma, \overrightarrow{R})$.
- If $op = collapse$ and $top_1(s) = \gamma \in \Gamma_{\mathrm{rk}}$ then Éloïse plays $(q, rew_1^\alpha; collapse)$.
- If $op = collapse$ and $top_1(s) = (\gamma, \overrightarrow{R})$ then Éloïse plays $(\lightning, id)$. We shall later see that this move is always valid.

- If $op = push_1^{\beta,n}$ then Éloïse plays plays $(q^\beta, rew_1^\alpha; id)$ if $top_1(s) = \gamma$ and she plays $(q^\beta, rew_1^{(\alpha,\overrightarrow{R})}; id)$ if $top_1(s) = (\gamma, \overrightarrow{R})$.

In this last case, or in the case where $p \in Q_{\mathbf{A}}$ and Abelard plays some $(q^\beta, rew_1^\alpha; id)$ (*resp.* some $(q^\beta, rew_1^{(\alpha,\overrightarrow{R})}; id)$), we also have to explain how Éloïse behaves from $(q^\beta, rew_1^\alpha(s))$ (*resp.* $(q^\beta, rew_1^{(\alpha,\overrightarrow{R})}(s))$.

Éloïse has to play $(q^?, push_1^{(\beta,\overrightarrow{S}),1})$ where $\overrightarrow{S} \in (2^{Q_{\mathrm{rk}}})^{d+1}$ describes which states can be reached if the n-link created by pushing β (or a copy of it) is used for collapsing the stack, depending on the smallest visited colour in the meantime. In order to define $\overrightarrow{S}$, she considers the set of all possible continuations of $\Lambda_{\mathrm{rk}} \cdot (q, push_1^{\beta,n}(\sigma))$ (where (p, σ) denotes the last vertex of Λ_{rk}) where she respects her strategy Φ_{rk}. For each such play, she checks whether some configuration of the form $(r, pop_n(\sigma))$ is eventually reached by collapsing (possibly a copy of the) n-link created by $push_1^{\beta,n}$. If such an r exists, she considers the smallest colour i visited from the moment where the link was created to the moment $collapse$ is performed (*i.e.* the link rank just before collapsing). For every $i \in \{0, \dots d\}$, the set S_i is defined to be the set of states $r \in Q$ such that the preceding case happens. Formally,

$$S_i = \{r \mid \exists\ \Lambda_{\mathrm{rk}} \cdot v_0 \cdots v_k \cdot v_{k+1} \cdots \text{ play in } \mathbb{G}_{\mathrm{rk}} \text{ where Éloïse respects } \Phi_{\mathrm{rk}} \text{ and s.t.}$$
$$v_0 = (q, push_1^{\beta,n}(\sigma)),\ v_{k+1} = (r, pop_n(\sigma)) \text{ is obtained by applying } collapse \text{ from } v_k,$$
$$v_0 \text{ is the link ancestor of } v_k \text{ and } i \text{ is the link rank in } v_k\}$$

Finally, we set $\overrightarrow{S} = (S_0, \dots, S_d)$ and Éloïse plays $(q^?, push_1^{(\beta,\overrightarrow{S}),1})$.

Update of Λ_{rk}. The memory Λ_{rk} is updated after each visit to a configuration with a control state in $Q_{\mathrm{rk}} \cup \{tt, ff\}$. We have several cases depending on the transition.

- If the last transition is of the form $(q, rew_1^\alpha; op)$ or $(q, rew_1^{(\alpha,\overrightarrow{R})}; op)$ with op being neither of the form $push_1^{\beta,n}$ nor $collapse$, then we extend Λ_{rk} by applying transition $(q, rew_1^\alpha; op)$, *i.e.* if (p, σ) denotes the last configuration in Λ_{rk}, then the updated memory is $\Lambda_{\mathrm{rk}} \cdot (q, op(rew_1^\alpha(\sigma)))$.
- If the last transition is of the form (tt, id) or (ff, id), the play is in a sink configuration. Therefore we do not update Λ_{rk} as the play will loop forever.
- If the last transitions form a sequence of the form $(q^\beta, rew_1^\alpha; id) \cdot (q^?, push_1^{(\beta,\overrightarrow{S}),1}) \cdot (q, id)$ or of the form $(q^\beta, rew_1^{(\alpha,\overrightarrow{R})}; id) \cdot (q^?, push_1^{(\beta,\overrightarrow{S}),1}) \cdot (q, id)$, then the updated memory is $\Lambda_{\mathrm{rk}} \cdot (q, push_1^{\beta,n}(\sigma))$, where (p, σ) denotes the last configuration in Λ_{rk}.
- If the last transitions form a sequence of the form $(q^\beta, rew_1^\alpha; id) \cdot (q^?, push_1^{(\beta,\overrightarrow{S}),1}) \cdot (r^i, id) \cdot (r, pop_n)$ or of the form $(q^\beta, rew_1^{(\alpha,\overrightarrow{R})}; id) \cdot (q^?, push_1^{(\beta,\overrightarrow{S}),1}) \cdot (r^i, id) \cdot (r, pop_n)$, then we extend Λ_{rk} by a sequence of actions (consistent with Φ_{rk}) that starts by performing transition $(q, push_1^{\beta,n})$ and ends up by collapsing (possibly a copy of) the link created at this first step and goes to state p while visiting i as a minimal colour in the meantime. By definition of $\overrightarrow{S}$ such a sequence always exists. More formally, if (p, σ) denotes the last configuration in Λ_{rk}, then the updated memory is a play in $\mathbb{G}_{\mathrm{rk}}$, $\Lambda_{\mathrm{rk}} \cdot v_0 \cdots v_k \cdot v_{k+1}$, where Éloïse respects Φ_{rk} and such that $v_0 = (q, push_1^{\beta,n}(\sigma))$, $v_{k+1} = (r, pop_n(\sigma))$ is obtained by applying $collapse$ from v_k, v_0 is the link ancestor of v_k and i is the link rank in v_k.

Therefore, with any partial play Λ_{lf} in $\mathbb{G}_{\mathrm{lf}}$ in which Éloïse respects her strategy Φ_{lf}, is associated a partial play Λ_{rk} in $\mathbb{G}_{\mathrm{rk}}$. An immediate induction shows that Λ_{rk} is a play where Éloïse respects Φ_{rk}. The same arguments works for any infinite play Λ_{lf} that does not contain a state in $\{tt, ff\}$, and the corresponding play Λ_{rk} is therefore infinite, starts from $\nu(p_0, s_0)$ and Éloïse respects Φ_{rk} in that play. Therefore it is a winning play.

Moreover, if Λ_{lf} is an infinite play that does not contain a state in $\{tt, ff\}$, it easily follows from the definitions of Φ_{lf} and Λ_{rk} that the smallest infinitely visited colour in Λ_{lf} is the same as the one in Λ_{rk}. Hence, any infinite play in $\mathbb{G}_{\mathrm{lf}}$ starting from $\nu(p_0, s_0)$ where Éloïse respects Φ_{lf} and that does not contain a state in $\{tt, ff\}$ is won by Éloïse.

Now, consider a play that contains a state in $\{tt, ff\}$ (hence loops on it forever). Reaching a configuration with state in $\{tt, ff\}$ is necessarily by simulating a $collapse$ from some configuration with a top_1-element of the form $(\alpha, \overrightarrow{R})$. We should distinguish between those elements $(\alpha, \overrightarrow{R})$ that are "created" before (*i.e.* by the ν function) or during the play (by Éloïse). For the second ones, one may note that whenever Éloïse wants to simulate a collapse, she can safely goes to state tt (meaning Φ_{lf} is well defined): indeed, if this was not the case, it would contradict the way $\overrightarrow{S}$ was defined when simulating the original creation of the link. For the same reason, Abelard can never reach state ff provided Éloïse respects her strategy Φ_{lf}. Now consider an element $(\alpha, \overrightarrow{R})$ created by ν and assume that one player wants to simulate a collapse from some

configuration with such a top_1-element. Call $\Lambda_{\rm lf}$ the partial play just before and call $\Lambda_{\rm rk}$ the associated play in $\mathbb{G}_{\rm rk}$. Then in $\Lambda_{\rm rk}$, Éloïse respects her winning strategy $\Phi_{\rm rk}$. If she has to play next in $\Lambda_{\rm rk}$, strategy $\Phi_{\rm rk}$ indicates to play $collapse$; if it is Abelard's turn to move he can play $collapse$. In both case, the configuration that is reached after collapsing is winning for Éloïse (it is a configuration visited in a winning play). Hence, by definition of ν, its control state belongs to R where $\overrightarrow{R} = (R, \cdots, R)$, and therefore from the current vertex in $\mathbb{G}_{\rm lf}$ there are no transition to ff and there is at least one to $\sharp$. Therefore plays where Éloïse respects $\Phi_{\rm lf}$ and that contain a state in $\{\sharp, ff\}$ necessarily contains state $\sharp$ hence are won by Éloïse.

Altogether, it proves that $\Phi_{\rm lf}$ is a winning strategy for Éloïse in $\mathbb{G}_{\rm lf}$ from $\nu(v_0)$.

Let us now prove the converse implication Assume that the configuration $\nu(p_0, s_0)$ is winning for Éloïse in $\mathbb{G}_{\rm lf}$, and let $\Phi_{\rm lf}$ be a winning strategy for her. Using $\Phi_{\rm lf}$, we define a strategy $\Phi_{\rm rk}$ for Éloïse in $\mathbb{G}_{\rm rk}$ from (p_0, s_0). First, recall how $\nu(p_0, s_0)$ is defined: every symbol α in s with an n-link is replaced by a pair $(\alpha, (R, \ldots, R))$ where R is the set of states r such that Éloïse wins from (r, s') where s' is the stack obtained by first removing every symbol (and stacks) above α and then performing a $collapse$. We can therefore assume that we have a collection of winning strategies, one per each such configurations (r, s') – call such a strategy $\Phi_{\rm rk}^{r,s'}$. Then, during a play where Éloïse respects $\Phi_{\rm rk}$, if one eventually visits such a configuration (r, s'), the strategy $\Phi_{\rm rk}$ will mimic the winning strategy $\Phi_{\rm rk}^{r,s'}$ from that point and therefore the resulting play will be winning for Éloïse. Then in the rest of this description we mostly focus on the case of plays where this phenomenon is not happening.

The strategy $\Phi_{\rm rk}$ maintains as a memory a partial play $\Lambda_{\rm lf}$ in $\mathbb{G}_{\rm lf}$, that is an element in $V_{\rm lf}^*$ (where $V_{\rm lf}$ denotes the set of vertices of $G_{\rm lf}$). At the beginning $\Lambda_{\rm lf}$ is initialised to the configuration $\nu(p_0, s_0)$. After having played $\Lambda_{\rm rk}$, the play $\Lambda_{\rm lf}$ will satisfy the following invariant. Assume that the play $\Lambda_{\rm lf}$ ends in a configuration (p, s) then the following holds.

- If $top_1(s) = \alpha$, the last configuration of $\Lambda_{\rm rk}$ has control state p and its top_1-element is α and it has a k-link for some $k < n$.
- If $top_1(s) = (\alpha, \overrightarrow{R})$, the last configuration of $\Lambda_{\rm rk}$ has control state p and its top_1-element is α and it has an n-link. Moreover, if Éloïse keeps respecting $\Phi_{\rm rk}$ in the rest of the play, if (possibly a copy of) this link is eventually used in a $collapse$, then the state that will be reached just after doing the $collapse$ will belong to R_i where i will be the link rank just before collapsing.

We first describe $\Phi_{\rm rk}$ and we then explain how $\Lambda_{\rm lf}$ is updated. Recall that we switch to a known winning strategy in case we do a $collapse$ from (possibly a copy of) an n-link that was already in s_0.

Choice of the move. Assume that the play is in some vertex (p, s) with $p \in Q_{{\rm rk},\mathbf{E}}$. The move given by $\Phi_{\rm rk}$ depends on $\Phi_{\rm lf}(\Lambda_{\rm lf}) = (q, rew; op)$ (we shall later argue that $\Phi_{\rm lf}$ is well defined while proving that it is winning).

- If $q \in Q_{\rm rk}$ then Éloïse plays $(q, rew_1^\alpha; op)$ where α is such that either $rew = rew_1^\alpha$ or $rew = rew_1^{(\alpha, \overrightarrow{R})}$. Note that in this case, op is neither a $collapse$ involving an n-link nor of the form $push_1^{\beta,n}$.
- If $q = r^\beta$ then Éloïse plays to $(r, rew_1^\alpha; push_1^{\beta,n})$ where α is such that either $rew = rew_1^\alpha$ or $rew = rew_1^{(\alpha, \overrightarrow{R})}$.
- If $q = \sharp$ then Éloïse plays $(r, collapse)$ for some arbitrary $r \in R_i$ where we let $i = LinkRk(p, top_1(s))$ and $(\alpha, \overrightarrow{R})$ denotes the top_1-element of the last vertex of $\Lambda_{\rm lf}$. Note that in this case, the collapse involves an n-link.

Update of $\Lambda_{\rm lf}$. The memory $\Lambda_{\rm lf}$ is updated after each move (played by any of the two players). We have several cases depending on the last transition.

- If the last transition is of the form $(q, rew_1^\alpha; op)$ and op is neither a $collapse$ involving an n-link nor of the form $push_1^{\beta,n}$, then $\Lambda_{\rm lf}$ is extended by mimicking the same transition, *i.e.* if (p, σ) denotes the last configuration in $\Lambda_{\rm lf}$, then the updated memory is $\Lambda_{\rm lf} \cdot (q, op(rew_1^\alpha(\sigma))$ if $top_1(\sigma) = \gamma$ for some $\gamma \in \Gamma_{\rm rk}$, and is $\Lambda_{\rm lf} \cdot (q, op(rew_1^{(\alpha, \overrightarrow{R})}(\sigma))$ if $top_1(\sigma) = (\gamma, \overrightarrow{R})$ for some $(\gamma, \overrightarrow{R}) \in \Gamma_{\rm lf}$.
- If the last transition is of the form $(q, rew_1^\alpha; push_1^{\beta,n})$ then, we let (p, σ) denotes the last configuration in $\Lambda_{\rm lf}$.
  If $top_1(\sigma) = \gamma$ for some $\gamma \in \Gamma_{\rm rk}$ then the updated memory $\Lambda_{\rm lf} \cdot (q^\beta, rew_1^\alpha(\sigma)) \cdot (q^?, push_1^{(\beta, \overrightarrow{R}),1}(rew_1^\alpha(\sigma))) \cdot (q, push_1^{(\beta, \overrightarrow{R}),1}(rew_1^\alpha(\sigma)))$ where $\Phi_{\rm lf}(\Lambda_{\rm lf} \cdot (q^\beta, rew_1^\alpha(\sigma))) = (q^?, push_1^{(\beta, \overrightarrow{R}),1}(rew_1^\alpha(\sigma)))$.
  If $top_1(\sigma) = (\gamma, \overrightarrow{S})$ for some $(\gamma, \overrightarrow{S}) \in \Gamma_{\rm lf}$ then the updated memory $\Lambda_{\rm lf} \cdot (q^\beta, rew_1^{(\alpha, \overrightarrow{S})}(\sigma)) \cdot (q^?, push_1^{(\beta, \overrightarrow{R}),1}(rew_1^{(\alpha, \overrightarrow{S})}(\sigma))) \cdot (q, push_1^{(\beta, \overrightarrow{R}),1}(rew_1^{(\alpha, \overrightarrow{S})}(\sigma)))$ where $\Phi_{\rm lf}(\Lambda_{\rm lf} \cdot (q^\beta, rew_1^{(\alpha, \overrightarrow{S})}(\sigma))) = (q^?, push_1^{(\beta, \overrightarrow{R}),1}(rew_1^{(\alpha, \overrightarrow{S})}(\sigma)))$.

- If the last transition is of the form $(r, collapse)$ and the $collapse$ follows to an n-link, then we have two cases. Either the $collapse$ was following (possibly a copy of) an n-link that was already in s_0 in case we claim (and prove later) that one ends up in a winning configuration and then switch to a corresponding winning strategy as already explained. Either one follows an n-link that was created during the play, in which case we let $\Lambda_{\rm lf} = v_0 \cdots v_m$ and denote by v_i the link ancestor of v_m [10]. Then the updated memory is obtained by backtracking inside $\Lambda_{\rm lf}$ until reaching the configuration where the (simulation of the) collapsed n-link was created (this configuration is v_i, the link ancestor) and then extend it by a choice of Abelard consistent with the $collapse$. That is the updated memory is $v_0 \cdots v_i \cdot (r^\ell, \sigma) \cdot (r, pop_n(\sigma))$ where $v_i = (p^?, \sigma)$ and ℓ denotes the link rank in the configuration $\Lambda_{\rm rk}$ was just before doing the $collapse$.

Therefore, with any partial play $\Lambda_{\rm rk}$ in $\mathbb{G}_{\rm rk}$ in which Éloïse respects her strategy $\Phi_{\rm rk}$, is associated a partial play $\Lambda_{\rm lf}$ in $\mathbb{G}_{\rm lf}$. Note that if we end up in a configuration that is known to be winning, $\Lambda_{\rm lf}$ is no longer extended. This also implies that when collapsing an n-link that was already in s_0 one necessarily ends up in a winning configuration. Indeed assume the contrary and let $\Lambda_{\rm lf}$ be the constructed play before collapsing: then either Éloïse has to play and therefore moves to tt (and therefore the configuration in $\Lambda_{\rm rk}$ after collapsing is winning by definition of ν, leading a contradiction) of Abelard could move to ff (leading a contradiction with $\Phi_{\rm lf}$ being winning). Therefore from now on we restrict our attention to the case where the n-links (and their copies) in s_0 are never used to do a $collapse$.

An easy induction shows that Éloïse respects $\Phi_{\rm lf}$ in $\Lambda_{\rm lf}$. The same arguments works for an infinite play $\Lambda_{\rm rk}$, and the corresponding play $\Lambda_{\rm lf}$ is therefore infinite (one simply considers the limit of the $\Lambda_{\rm lf}$ in the usual way [11]), starts from $\nu(p_0, s_0)$, never visits a state in $\{tt, ff\}$ and Éloïse respects $\Phi_{\rm lf}$ in that play. Therefore it is a winning play.

Now, in order to conclude that any play $\Lambda_{\rm rk}$ in $\mathbb{G}_{\rm rk}$ in which Éloïse respects strategy $\Phi_{\rm rk}$ is winning for her, one needs to relate the sequence of colours in $\Lambda_{\rm rk}$ with the one in $\Lambda_{\rm lf}$. For this, we introduce a notion of factorisation of a partial play $\Lambda_{\rm rk} = v_0 v_1 \cdots v_m$ in $\mathbb{G}_{\rm rk}$ (we should later note that it directly extends to infinite plays). A factor is a nonempty sequence of vertices of the following kind:

(1) it is a sequence $v_h \cdots v_k$ such that the stack operation from v_{h-1} to v_h is of the form $rew_1^\alpha; push_1^{n,\beta}$, the stack operation from v_{k-1} to v_k is a $collapse$ involving an n-link, and v_h is the link ancestor of v_k.
(2) or it is a single vertex;

Then the factorisation of $\Lambda_{\rm rk}$ denoted $Fact(\Lambda_{\rm rk})$ is a sequence of factors inductively defined as follows (we underline factors to make them explicit): $Fact(\Lambda_{\rm rk}) = \underline{v_0 \cdots v_k}, Fact(v_{n+1} \cdots v_n)$ if there exists some k such that $v_0 \cdots v_k$ is as in (1) above, and $Fact(\Lambda_{\rm rk}) = \underline{v_0}, Fact(v_1 \cdots v_n)$ otherwise.

In the following, we refer to the *colour* of a factor as the minimal colour of its elements.

Note that the previous definition is also valid for infinite plays. Now we easily get the following proposition (the result is obtained by reasoning on partial play using a simple induction combined with a case analysis. Then it directly extends to infinite plays).

Proposition 6. *Let $\Lambda_{\rm rk}$ be some infinite play in $\mathbb{G}_{\rm rk}$ starting from (p_0, s_0) where Éloïse respects $\Phi_{\rm rk}$ and assume that there is no $collapse$ that follows (possibly a copy of) an n-link already in s_0. Let $\Lambda_{\rm lf}$ be the associated infinite play in $\mathbb{G}_{\rm lf}$ constructed from $\Phi_{\rm rk}$. Let $\Lambda_{{\rm rk},0}, \Lambda_{{\rm rk},1}, \cdots$ be the factorisation of $\Lambda_{\rm rk}$ and, for every $i \geqslant 0$, let c_i be the colour of $\Lambda_{{\rm rk},i}$.*

Then the sequence $(c_i)_{i \geqslant 0}$ and the sequence of colours visited in $\Lambda_{\rm lf}$ have the same $\liminf$.

The previous proposition directly implies that $\Phi_{\rm rk}$ is a winning strategy for Éloïse from (p_0, s_0) in $\mathbb{G}_{\rm rk}$.

In order to complete the proof of Theorem 7 it remains to establish the following proposition.

Proposition 7. *If there is an n-CPDA transducer $\mathcal{S}_{\rm lf}$ synchronised with $\mathcal{A}_{\rm lf}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\rm lf}$ from $\nu(q_{0,\rm rk}, \bot_n)$, then one can effectively construct an n-CPDA transducer $\mathcal{S}_{\rm rk}$ synchronised with $\mathcal{A}_{\rm rk}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\rm rk}$ from the initial configuration $(q_{0,\rm rk}, \bot_n)$.*

Proof: The result follows from a carefully analysis of how we defined $\Phi_{\rm rk}$ from $\Phi_{\rm lf}$ in the proof of Theorem 8. As we now only focus on the initial configuration $(q_{0,\rm rk}, \bot_n)$ we will not have to deal with the special case of doing a $collapse$ following (possibly a copy of) an n-link originally in the initial configuration. Also note that $\nu(q_{0,\rm rk}, \bot_n) = (q_{0,\rm rk}, \bot_n)$.

[10] Here we implicitly extends the notion of link ancestor as follows. In $\mathbb{G}_{\rm lf}$ instead of creating n-link one pushes symbol of the form $(\beta, \overrightarrow{R})$: hence whenever doing a $push_1^{(\beta,\overrightarrow{R}),1}$ one attaches to the vector $\overrightarrow{R}$ the index of the current configuration. Then if the top_1 element of v_n is some $(\beta, \overrightarrow{R})$ then the link ancestor of v_m is defined to be v_i where i is the indexed attached with $\overrightarrow{R}$. Note in particular that the control state in the link ancestor is necessarily of the form $p^?$.

[11] Let $(u_n)_{n \geqslant 0}$ be a sequence of finite words. For any $n \geqslant 0$ let $u_n = u_n^0 \cdots u_n^{k_n}$. Then the limit of the sequence $(u_n)_{n \geqslant 0}$ is the (possibly infinite) word $\alpha = \alpha^0 \alpha^1 \cdots$ such that α is maximal for the prefix ordering and for all $0 \leqslant i < |\alpha|$ there is some N_i such that $u_n^i = \alpha^i$ for all $n \geqslant N_i$.

In our setting, the play $\Lambda_{\rm lf}$ associated with an infinite play $\Lambda_{\rm rk}$ is defined as the limit of the sequence of partial plays $(\Lambda_{\rm lf}^n)_{n \geqslant 0}$ where $\Lambda_{\rm lf}^n$ is the partial play associated with $\Lambda_{\rm rk}$ truncated to its $n+1$ first vertices. From the definitions of the $\Lambda_{\rm lf}^n$ it is easily verified that the limit $\Lambda_{\rm lf}$ is infinite.

Recall that $\Phi_{\rm rk}$ uses as a memory a partial play $\Lambda_{\rm lf}$ in $\mathbb{G}_{\rm lf}$ and considers the value of $\Phi_{\rm lf}(\Lambda_{\rm lf})$ to determine the next move to play. Now assume that $\Phi_{\rm lf}$ is realised by an n-CPDA transducer $\mathcal{S}_{\rm lf}$ synchronised with $\mathcal{A}_{\rm lf}$. Hence, instead of storing $\Lambda_{\rm lf}$ it suffices to store the configuration $\mathcal{S}_{\rm lf}$ is in after reading $\Lambda_{\rm lf}$.

One can also notice that the stack $s_{\rm rk}$ in the last configuration of some partial play $\Lambda_{\rm rk}$ and the stack $s_{\rm lf}$ in the last configuration of the associated $\Lambda_{\rm lf}$ have the same shapes *provided* one replaces in $s_{\rm lf}$ every 1-link from a symbol in $\Gamma_{\rm rk} \times (2^{Q_{\rm rk}})^{d+1}$ by an n-link. Recall that these 1-links are never used to perform a $collapse$: hence replacing those 1-links by n-links does not change the issue of the game, and if one does a similar transformation on $\mathcal{S}_{\rm lf}$ it still realises a winning strategy, and it is synchronised with the transformed version of $\Lambda_{\rm lf}$.

Now, it follows from the way one defined $\Phi_{\rm rk}$ (both the choice of the move and the memory update) that one can design an n-CPDA transducer $\mathcal{S}_{\rm rk}$ synchronised with $\mathcal{A}_{\rm rk}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\rm rk}$ from the initial configuration $(q_{0,\rm rk}, \bot_n)$. In all cases but one $\mathcal{S}_{\rm rk}$ simulates $\mathcal{S}_{\rm lf}$. The only problematic case is when the move to play is some $(r, collapse)$ involving an n-link. Indeed, one needs to backtrack in $\Lambda_{\rm lf}$ (namely retrieve the configuration of $\mathcal{S}_{\rm lf}$ after the link ancestor) and extend it by doing (r^ℓ, id) (where ℓ is the link rank) and then (r, pop_n); one needs to retrieve the configuration of $\mathcal{S}_{\rm lf}$ right after this. If one performs a $collapse$ in $\mathcal{S}_{\rm rk}$, one directly retrieves the stack content, but the control state of $\mathcal{S}_{\rm lf}$ is still missing. However, one can modify $\mathcal{S}_{\rm lf}$ so that after the simulation of the creation of an n-link, *i.e.* after a symbol of the form $(\gamma, \overrightarrow{R})$ in pushed, it stores in its top_1-element the control state it will be in after doing the transitions $(r^\ell, id)(r, pop_n)$, for each $0 \leqslant \ell \leqslant d$ and each $r \in R_\ell$ (this can easily be computed). As this information is then propagated when copying the symbol/link, it is available in the top_1-element before doing a $collapse$ involving an n-link, hence $\mathcal{S}_{\rm rk}$ can also correctly retrieve the control state of $\mathcal{S}_{\rm lf}$.

From this (somehow informal) description of $\mathcal{S}_{\rm rk}$ the reader should be convinced that $\mathcal{S}_{\rm rk}$ correctly simulates $\mathcal{S}_{\rm lf}$ on $\Lambda_{\rm lf}$, hence realises a winning strategy in $\mathbb{G}_{\rm rk}$. The fact that $\mathcal{S}_{\rm rk}$ is synchronised with $\mathcal{A}_{\rm rk}$ follows from the fact that it is synchronised with the variant of $\mathcal{S}_{\rm lf}$ that itself is synchronised with the variant of $\Lambda_{\rm lf}$ which is synchronised with $\Lambda_{\rm rk}$. ■

Opitmisation. The set $Q_{\rm lf}$ has size $\mathcal{O}(|Q_{\rm rk}|(|\Gamma_{\rm rk}| + d))$, which is not very satisfactory for complexity reasons. Actually, one would prefer a variant of the construction where $|\Gamma_{\rm rk}|$ does not appear in the blowup concerning states. This factor actually comes from states $\{q^\gamma \mid q \in Q_{\rm rk},\ \gamma \in \Gamma_{\rm rk}\}$, and one can easily get read of them by doing the following modification of $\mathcal{A}_{\rm lf}$. When simulating a $push_1^{\beta,n}$, instead of going to q^β, one stores β (thanks to a rew_1 operation) in the top_1 element of the stack (hence the stack alphabet gets augmented by a linear factor in $|\Gamma_{\rm rk}|$) and goes to a special state $q^!$. State $q^!$ is controlled by Éloïse and the transition function is the same as from q^β where β is the symbol stored on the top_1-element of the stack.

It is straightforward that this modification does not change the validity of the previous statements.

If we summarise, the overall blowup in the transformation from $\mathbb{G}_{\rm rk}$ to $\mathbb{G}_{\rm lf}$ given by Theorem 7 is as follows.

Proposition 8. *Let $\mathcal{A}_{\rm rk}$ and $\mathcal{A}_{\rm lf}$ be as in Theorem 7. Then the set of states of $\mathcal{A}_{\rm lf}$ has size $\mathcal{O}(|Q_{\rm rk}|d)$ and the stack alphabet of $\mathcal{A}_{\rm lf}$ has size $\mathcal{O}(|\Gamma_{\rm rk}|^2 \cdot 2^{|Q_{\rm rk}|(d+1)})$.*

Finally the set of colours used in $\mathbb{G}_{\rm rk}$ and $\mathbb{G}_{\rm lf}$ are the same.

Proof: By construction together with the optimisation below. ■

7) Reducing the Order:

In the previous section we have constructed from a game played on a *rank-aware* n-CPDP another game played on an n-CPDP that does not create n-links. The winning regions (*resp.* winning strategies realised by n-CPDA transducer) in the original game can then be recover from the winning regions (*resp.* winning strategies realised by n-CPDA transducer) in the latter game.

In this section, we prove a result in a similar flavour. Namely, starting from a game played on an n-CPDP that does not create n-links, we construct a game played on an $(n-1)$-CPDP, and we show that the winning regions(*resp.* winning strategies realised by n-CPDA transducer) in the original game can be recover from the winning regions (*resp.* winning strategies realised by $(n-1)$-CPDA transducer) in the latter game.

We situate the techniques developed here in a general and abstract framework of (order-1) pushdown automata whose stack alphabet is a *possibly infinite* set: *abstract pushdown automata.* We start by introducing this concept and show how n-CPDP that does not create n-links fit into it. Then, we introduce the notion of *conditional games*. Finally, we show how such games can be solved by reduction to a $(n-1)$-CPDP parity game, and from the proof we also get the expect result on the existence of strategies realised by CPDA transducers.

We situate the techniques developed here in a general and abstract framework of (order-1) pushdown automata whose stack alphabet is a *possibly infinite* set.

An *abstract pushdown automaton* is a tuple $\mathcal{A} = \langle A, Q, \Delta, q_0 \rangle$ where A is a (possibly infinite) set called an *abstract pushdown alphabet* and containing a bottom-of-stack symbol denoted $\bot \in A$, Q is a finite set of states, $q_0 \in Q$ is an initial state and

$$\Delta : Q \times A \to 2^{Q \times A^{\leqslant 2}}$$

is the transition relation (here $A^{\leqslant 2} = \{\varepsilon\} \cup A \cup A \cdot A$ are the words over A of length at most 2). We additionally require that for all $a \neq \bot$, $\Delta(q, a)$ does not contain any element of the form $(q, a\bot)$ nor $(q, \bot a)$, and that $\Delta(q, \bot)$ does not contain any element of the form (q', ε) nor (q', a) nor (q, ab) with $a \neq \bot$ or $b = \bot$, *i.e.* the bottom-of-stack symbol can only occur at the bottom of the stack, and is never popped nor rewritten.

An *abstract pushdown content* is a word in $St = \bot(A \backslash \{\bot\})^*$. A configuration of $\mathcal{A}$ is a pair (q, u) with $q \in Q$ and $u \in St$.

Remark 9. *In general an abstract pushdown automaton is not finitely describable, as the domain of Δ is infinite and no further assumption is made on Δ.*

Example 22. *An order-1 pushdown process is an abstract pushdown automaton whose stack alphabet is finite.*

A abstract pushdown automaton $\mathcal{A}$ induces a possibly infinite graph, called an *abstract pushdown graph,* denoted $G = (V, E)$, whose vertices are the configurations of $\mathcal{A}$ and edges are defined by the transition relation Δ, *i.e.*, from a vertex $(q, u \cdot a)$ one has an edge to $(q', u \cdot u')$ whenever $(q', u') \in \Delta(q, a)$.

Example 23. *Order-n CPDP that does not create n-links (*i.e. *never use stack operation of the form $push_1^{\gamma,n}$) are special cases of abstract pushdown automata. Indeed, let $n > 1$ and consider such an order-n CPDP $\mathcal{A} = \langle \Gamma, Q, \Delta, q_0 \rangle$. Let A be the set of all order-$(n-1)$ stacks over Γ, and for every $p \in Q$ and $a \in A$ with $\gamma = top_1(a)$, we define $\Delta'(p, a)$ by*

- $(q, \varepsilon) \in \Delta'(p, a)$ *iff* $(q, pop_n) \in \Delta(q, \gamma)$*;*
- $(q, a' \cdot a') \in \Delta'(p, a)$ *with* $a' = rew_1^{\alpha}(a)$ *iff* $(q, rew_1^{\alpha}; push_n) \in \Delta(q, \gamma)$*;*
- $(q, a') \in \Delta'(p, a)$ *with* $a' = op(rew_1^{\alpha}(a))$ *iff* $(q, rew_1^{\alpha}; op) \in \Delta(q, \gamma)$ *and* $op \notin \{pop_n, push_n\}$.

It follows that $\mathcal{A}$ and the abstract pushdown automaton $\langle A, Q, \Delta', q_0 \rangle$ have isomorphic transition graphs.

Consider now a partition $Q_{\mathbf{E}} \cup Q_{\mathbf{A}}$ of Q between Éloïse and Abelard. It induces a natural partition $V_{\mathbf{E}} \cup V_{\mathbf{A}}$ of V by setting $V_{\mathbf{E}} = Q_{\mathbf{E}} \times St$ and $V_{\mathbf{A}} = Q_{\mathbf{A}} \times St$. The resulting game graph $\mathcal{G}_{\text{abs}} = (V_{\mathbf{E}}, V_{\mathbf{A}}, E)$ is called an *abstract pushdown game graph*. Let Ω be a colouring function from Q to a finite set of colours $C \subset \mathbb{N}$. This function is easily extended to a function from V to C by setting $\Omega((q, \sigma)) = \Omega(q)$. Finally, an *abstract pushdown parity game* is a parity game played on such an abstract pushdown game graph where the colouring function is defined as above.

For every subset $R \subseteq Q$ we define the *conditional game induced by R over $\mathcal{G}_{\text{abs}}$*, denoted $\mathbb{G}_{\text{abs}}(R)$, as the game played over $\mathcal{G}_{\text{abs}}$ where a play Λ is winning for Éloïse iff one of the following happens:

- In Λ no configuration with an empty stack (*i.e.* of the form $(q, \bot)$) is visited, and Λ satisfies the parity condition.
- In Λ a configuration with an empty stack is visited and the control state in the first such configuration belongs to R.

More formally, the set of winning plays $\mathcal{W}(R)$ in $\mathbb{G}_{\text{abs}}(R)$ is defined as follows ($\mathcal{W}_{par}$ stands for the parity condition on $\mathcal{G}_{\text{abs}}$):

$$\mathcal{W}(R) = [\mathcal{W}_{par} \backslash V^*(Q \times \{\bot\})V^{\omega}] \ \cup \ V^*(R \times \{\bot\})V^{\omega}$$

For any state q, any stack letter $a \neq \bot$, and any subset $R \subseteq Q$ it follows from Martin's Determinacy theorem [A5] that either Éloïse or Abelard has a winning strategy from $(q, \bot a)$ in $\mathbb{G}_{\text{abs}}(R)$. We denote by $\mathcal{R}(q, a)$ the set of subsets R for which Éloïse wins in $\mathbb{G}_{\text{abs}}(R)$ from $(q, \bot a)$:

$$\mathcal{R}(q, a) = \{R \subseteq Q \mid (q, \bot a) \text{ is winning for Éloïse in } \mathbb{G}_{\text{abs}}(R)\}$$

We now build a new game whose winning region embeds all the information needed to determine the sets $\mathcal{R}(q, a)$. Moreover in the underlying game graph the vertices no longer encode stacks.

For an infinite play $\Lambda = v_0 v_1 \cdots$ in $\mathbb{G}_{\text{abs}}$, let $Steps_{\Lambda}$ be the set of indices of positions where no configuration of strictly smaller stack height is visited later in the play. More formally, $Steps_{\Lambda} = \{i \in \mathbb{N} \mid \forall j \geqslant i \ sh(v_j) \geqslant sh(v_i)\}$, where $sh((q, \bot a_1 \cdots a_n)) = n + 1$. Note that $Steps_{\Lambda}$ is always infinite and hence induces a decomposition of the play Λ into finite pieces.

In the decomposition induced by $Steps_{\Lambda}$, a factor $v_i \cdots v_j$ is called a *bump* if $sh(v_j) = sh(v_i)$, called a *Stair* otherwise (that is, if $sh(v_j) = sh(v_i) + 1$ and $j = i + 1$).

For any play Λ with $Steps_{\Lambda} = \{n_0 < n_1 < \cdots\}$, we can define the sequence $(mcol_i^{\Lambda})_{i \geqslant 0} \in \mathbb{N}^{\mathbb{N}}$ by letting $mcol_i^{\Lambda} = \min\{\Omega(v_k) \mid n_i \leqslant k \leqslant n_{i+1}\}$. This sequence fully characterises the parity condition.

Proposition 9. *For a play* Λ*,* $\Lambda \in \mathcal{W}_{par}$ *iff* $\liminf((mcol_i^\Lambda)_{i\geqslant 0})$ *is even.*

In the sequel, we build a new parity game $\widetilde{\mathbb{G}}$ over a new game graph $\widetilde{\mathcal{G}} = (\widetilde{V}, \widetilde{E})$. This new game *simulates* the abstract pushdown graph, in the sense that the sequence of visited colours during a *correct* simulation of some play Λ in $\mathbb{G}_{\mathrm{abs}}$ is exactly the sequence $(mcol_i^\Lambda)_{i\geqslant 0}$. Moreover, a play in which a player does not correctly simulate the abstract pushdown game is losing for that player. We will show how the winning region in $\widetilde{\mathbb{G}}$ allows us to compute the sets $\{a \in A \mid R \in \mathcal{R}(q,a)\}$.

Before providing a description of the game graph $\widetilde{\mathcal{G}}$, let us consider the following informal description of this simulation game. We aim at simulating a play in the abstract pushdown game from the initial configuration $(q_0, \bot)$. In $\widetilde{\mathcal{G}}$ we keep track of only the control state and the top stack symbol of the simulated configuration.

The interesting case is when the simulated play is in configuration with control state p and top stack symbol a, and the player owning p wants to perform transition $(q, a'b)$, *i.e.* go to state q, rewrite a into a' and push b on top of it. For every strategy of Éloïse, there is a certain set of possible (finite) prolongation of the play (consistent with her strategy) that will end with popping b (or actually a symbol into which b was rewritten in the meantime) from the stack. We require Éloïse to declare a vector $\overrightarrow{S} = (S_0, \ldots, S_d)$ of $(+1)$ subsets of Q, where S_i is the set of all states the game can be in after popping (possibly a rewriting of) b along those plays where in addition the smallest visited colour while (possibly a rewriting of) b was on the stack is i.

Abelard has two choices. He can continue the game by pushing b onto the stack and updating the state (we call this a *pursue move*). Otherwise, he can pick a set S_i and a state $s \in S_i$, and continue the simulation from that state s (we call this a *jump move*). If he does a pursue move, then he remembers the vector $\overrightarrow{S}$ claimed by Éloïse; if later on, a transition of the form $(s, \varepsilon) \in Q \times \{\varepsilon\}$ is simulated, the play goes into a sink state (either *tt* or *ff*) that is winning for Éloïse if and only if the resulting state is in S_θ where θ is the smallest colour seen in the current level (this information will be encoded in the control state, reset after each pursue move and updated after each jump move). If Abelard does a jump move to a state s in S_i, the currently stored value for θ is updated to $\min(\theta, i, \Omega(s))$, which is the smallest colour seen since the current stack level was reached.

There are extra edges to simulate transition of the form $(q, a') \in Q \times A$ where the top stack element and the value of θ are updated.

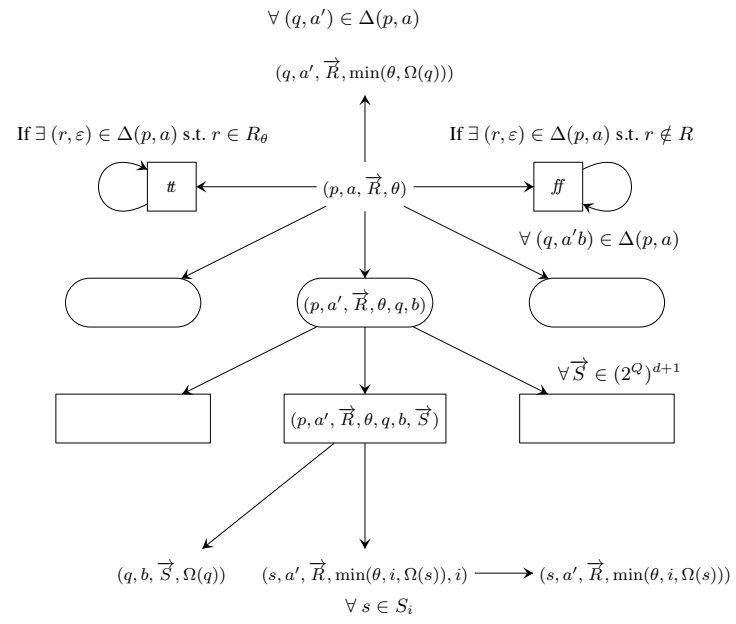


Figure 5. Local structure of $\widetilde{\mathcal{G}}$.

Let us now precisely describe the game graph $\widetilde{\mathcal{G}}$. We refer the reader to Figure 5.

- The main vertices of $\widetilde{\mathcal{G}}$ are those of the form $(p, a, \overrightarrow{R}, \theta)$, where $p \in Q$, $a \in A$, $\overrightarrow{R} = (R_0, \dots, R_d) \in (2^Q)^{d+1}$ and $\theta \in \{0, \dots, d\}$. A vertex $(p, a, \overrightarrow{R}, \theta)$ is reached when simulating a partial play Λ in $\mathbb{G}_{\text{abs}}$ such that:
  - The last vertex in Λ is (p, ua) for some $u \in A^*$.
  - Éloïse claims that she has a strategy to continue Λ in such a way that if a (or a rewriting of it) is eventually popped, the control state reached after popping belongs to R_i, where i is the smallest colour visited since the stack height was at least $|ua|$.
  - The colour θ is the smallest one since the current stack level was reached from a lower stack level.

  A vertex $(p, a, \overrightarrow{R}, \theta)$ is controlled by Éloïse if and only if $p \in Q_{\mathbf{E}}$.
- The vertices *ff* and *ff* are here to ensure that the vectors $\overrightarrow{R}$ encoded in the main vertices are correct. Both are sink vertices and are controlled by Abelard. Vertex *ff* gets colour 0 and vertex *ff* gets colour 1. As these vertices are sinks, a play reaching *ff* is won by Éloïse whereas a play reaching *ff* is won by Abelard.
  There is a transition from some vertex $(p, a, \overrightarrow{R}, \theta)$ to *ff*, if and only if there exists a transition rule $(r, \varepsilon) \in \Delta(p, a)$, such that $r \in R_\theta$ (this means that $\overrightarrow{R}$ is correct with respect to this transition rule). Dually, there is a transition from a vertex $(p, a, \overrightarrow{R}, \theta)$ to *ff* if and only if there exists a transition rule $(r, \varepsilon) \in \Delta(p, a)$ such that $r \notin R_\theta$ (this means that $\overrightarrow{R}$ is not correct with respect to this transition rule).
- To simulate a transition rule $(q, a') \in \Delta(p, a)$, the player that controls $(p, a, \overrightarrow{R}, \theta)$ moves to $(q, a', \overrightarrow{R}, \min(\theta, \Omega(q)))$. Note that the last component has to be updated as the smallest colour seen since the current stack level was reached is now $\min(\theta, \Omega(q))$.
- To simulate a transition rule $(q, a'b) \in \Delta(p, a)$, the player that controls $(p, a, \overrightarrow{R}, \theta)$ moves to $(p, a', \overrightarrow{R}, \theta, q, b)$. This vertex is controlled by Éloïse who has to give a vector $\overrightarrow{S} = (S_0, \dots, S_d) \in (2^Q)^{d+1}$ that describes the control states that can be reached if b (or a symbol that rewrites it later) is eventually popped. To describe this vector, she goes to the corresponding vertex $(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{S})$.
  Any vertex $(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{S})$ is controlled by Abelard who chooses either to simulate a bump or a stair. In the first case, he additionally has to pick the minimal colour of the bump. To simulate a bump with minimal colour i, he goes to a vertex $(s, a', \overrightarrow{R}, \min(\theta, i, \Omega(s)))$, for some $s \in S_i$, through an intermediate vertex $(s, a', \overrightarrow{R}, \min(\theta, i, \Omega(s)), i)$ coloured by i.
  To simulate a stair, Abelard goes to the vertex $(q, b, \overrightarrow{S}, \Omega(q))$.
  The last component of the vertex (that stores the smallest colour seen since the currently simulated stack level was reached) has to be updated in all those cases. After simulating a bump of minimal colour i, the minimal colour is $\min(\theta, i, \Omega(s))$. After simulating a stair, this colour has to be initialised (since a new stack level is simulated). Its value, is therefore $\Omega(q)$, which is the unique colour since the (new) stack level was reached.

The vertices of the form $(p, a, \overrightarrow{R}, \theta)$ get colours $\Omega(p)$. Intermediate vertices of the form $(p, a', \overrightarrow{R}, \theta, q, b)$ or $(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{S})$ get colours d.

The following theorem relates the winning region in $\widetilde{\mathbb{G}}$ with $\mathbb{G}_{\text{abs}}$ and the conditional games induced over $\mathcal{G}_{\text{abs}}$.

Theorem 9. *The following holds.*

1) *A configuration $(p_{in}, \bot)$ is winning for Éloïse in $\mathbb{G}_{\text{abs}}$ if and only if $(p_{in}, \bot, (\varnothing, \dots, \varnothing), \Omega(p_{in}))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*
2) *For every $q \in Q$, $a \in A$ and $R \subseteq Q$, $R \in \mathcal{R}(q, a)$ if and only if $(q, a, (R, \dots, R), \Omega(q))$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*

The rest of the section is devoted to the proof of Theorem 9. We mainly focus on the proof of the first item, the proof of the second one being a subpart of it. We start by introducing some useful concept and then prove both implications.

To help readability, we will use upper-case letters, *e.g.* Λ or Φ, to denote objects (plays, strategies…) in $\mathbb{G}_{\text{abs}}$, and lower-case letters, *e.g.* λ or φ, to denote objects in $\widetilde{\mathbb{G}}$.

Recall that for an infinite play $\Lambda = v_0 v_1 \cdots$ in $\mathbb{G}_{\text{abs}}$ Steps_Λ denote the set of indices of positions where no configuration of strictly smaller stack height is visited later in the play. More formally, $\mathit{Steps}_\Lambda = \{i \in \mathbb{N} \mid \forall j \geqslant i \ sh(v_j) \geqslant sh(v_i)\}$, where $sh((q, \bot a_1 \cdots a_n)) = n + 1$. Note that Steps_Λ is always infinite and hence induces a factorisation of the play Λ into finite pieces. Recall that for any play Λ with $\mathit{Steps}_\Lambda = \{n_0 < n_1 < \cdots\}$, we define the sequence $(mcol_i^\Lambda)_{i \geqslant 0} \in \mathbb{N}^{\mathbb{N}}$ by letting $mcol_i^\Lambda = \min\{\Omega(v_k) \mid n_i \leqslant k \leqslant n_{i+1}\}$.

Indeed, for any play Λ with $\mathit{Steps}_\Lambda = \{n_0 < n_1 < \cdots\}$, one can define the sequence $(\Lambda_i)_{i \geqslant 0}$ by letting $\Lambda_i = v_{n_i} \cdots v_{n_{i+1}}$. Note that each of the Λ_i is either a bump or a stair. In the later we designate $(\Lambda_i)_{i \geqslant 0}$ as the *rounds factorisation* of Λ.

For any play λ in $\widetilde{\mathbb{G}}$, a *round* is a factor between two visits through vertices of the form $(p, a, \overrightarrow{R}, \theta)$. We have the following possible forms for a round.

- The round is of the form $(p, a, \overrightarrow{R}, \theta)(q, a', \overrightarrow{R}, \theta)$ and corresponds therefore to the simulation of a transition (q, a'). We designate it as a *trivial bump*.
- The round is of the form $(p, a, \overrightarrow{R}, \theta)(p, a', \overrightarrow{R}, \theta, q, b)(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{S})$ $(s, a', \overrightarrow{R}, \min(\theta, i, \Omega(s)), i)(s, a', \overrightarrow{R}, \min(\theta, i, \Omega(s)))$ and corresponds therefore to the simulation of a transition $(q, a'b)$ pushing b followed by a sequence of moves that ends by popping b (or a rewriting of it). Moreover i is the smallest colour encountered while b (or other top stack symbol obtained by successively rewriting it) was on the stack. We designate it as a *(non-trivial) bump*.
- The round is of the form $(p, a, \overrightarrow{R}, \theta)(p, a', \overrightarrow{R}, \theta, q, b)(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{S})$ $(q, b, \overrightarrow{S}, \Omega(q))$ and corresponds therefore to the simulation of a transition $(q, a'b)$ pushing a symbol b leading to a new stack level which the play will never go below. We designate it as a *stair*.

We define the *colour* of a round as the smallest colour of the vertices in the round.

For any play $\lambda = v_0 v_1 v_2 \cdots$ in $\widetilde{\mathbb{G}}$, we consider the subset of indices corresponding to vertices of the form $(p, a, \overrightarrow{R}, \theta)$. More precisely:

$$Rounds_\lambda = \{n \mid v_n = (p, a, \overrightarrow{R}, \theta),\ p \in Q,\ a \in A, \overrightarrow{R} \in (2^Q)^{d+1},\ 0 \leqslant \theta \leqslant d\}$$

The set $Rounds_\lambda$ induces a natural factorisation of λ into rounds. Indeed, let $Rounds_\lambda = \{n_0 < n_1 < n_2 < \cdots\}$, then for all $0 \leqslant i < |Rounds_\lambda|$ we let $\lambda_i = v_{n_i} \cdots v_{n_{i+1}}$. We call the sequence $(\lambda_i)_{i \geqslant 0}$ the *round factorisation* of λ. For every $i \geqslant 0$, λ_i is a round and the first vertex in λ_{i+1} equals the last one in λ_i. Moreover, $\lambda = \lambda_1 \odot \lambda_2 \odot \lambda_3 \odot \cdots$, where $\lambda_i \odot \lambda_{i+1}$ denotes the concatenation of λ_i with λ_{i+1} without its first vertex.

In order to prove both implications of Theorem 9, we build from a winning strategy for Éloïse in one game a winning strategy for her in the other game. The main argument to prove that the new strategy is winning is to prove a correspondence between the factorisations of plays in both games.

Direct implication

Assume that the configuration $(p_{in}, \bot)$ is winning for Éloïse in $\mathbb{G}_{\text{abs}}$, and let Φ be a corresponding winning strategy for her.

Using Φ, we define a strategy φ for Éloïse in $\widetilde{\mathbb{G}}$ from $(p_{in}, \bot, (\varnothing, \ldots, \varnothing), \Omega(p_{in}))$. The strategy φ maintains as a memory a partial play Λ in $\mathbb{G}_{\text{abs}}$, that is an element in V_{abs}^* (where V_{abs} denotes the set of vertices of $\mathcal{G}_{\text{abs}}$). At the beginning Λ is initialised to the vertex $(p_{in}, \bot)$. We first describe φ, and then we explain how Λ is updated. Both the strategy φ and the update of Λ, are described for a round.

Choice of the move. Assume that the play is in some vertex $(p, a, \overrightarrow{R}, \theta)$ for $p \in Q_{\mathbf{E}}$. The move given by φ depends on $\Phi(\Lambda)$:

- If $\Phi(\Lambda) = (r, \varepsilon)$, then Éloïse goes to $\#$ (Proposition 10 will prove that this move is always possible).
- If $\Phi(\Lambda) = (q, a')$, then Éloïse goes to $(q, a'b, \overrightarrow{R}, \min(\theta, \Omega(q)))$.
- If $\Phi(\Lambda) = (q, a'b)$, then Éloïse goes to $(p, a', \overrightarrow{R}, \theta, q, b)$.

In this last case, or in the case where $p \in Q_{\mathbf{A}}$ and Abelard goes to $(p, a', \overrightarrow{R}, \theta, q, b)$, we also have to explain how Éloïse behaves from $(p, a', \overrightarrow{R}, \theta, q, b)$. She has to provide a vector $\overrightarrow{S} \in (2^Q)^{d+1}$ that describes which states can be reached if b (or its successors by top rewriting) is eventually popped, depending on the smallest visited colour in the meantime. In order to define $\overrightarrow{S}$, Éloïse considers the set of all possible continuations of $\Lambda \cdot (q, ua'b)$ (where (p, ua) denotes the last vertex of Λ) where she respects her strategy Φ. For each such play, she checks whether some configuration of the form (s, ua') is visited after $\Lambda \cdot (q, ua'b)$, that is if the stack level of b is eventually left. If it is the case, she considers the first configuration (s, ua') appearing after $\Lambda \cdot (q, ua'b)$ and the smallest colour i since b and (possibly) its successors by top-rewriting were on the stack. For every $i \in \{0, \ldots d\}$, S_i, is exactly the set of states $s \in Q$ such that the preceding case happens. More formally,

$$S_i = \{s \mid \exists\ \Lambda \cdot (q, ua'b) v_0 \cdots v_k (s, ua') \cdots \text{ play in } \mathbb{G}_{\text{abs}} \text{ where Éloïse respects } \Phi \text{ and}$$
$$\text{s.t. } |v_j| > |ua|,\ \forall j = 0, \ldots, k \text{ and } \min(\{\Omega(v_j) \mid j = 0, \ldots, k\} \cup \{\Omega(q)\}) = i\}$$

Finally, we let $\overrightarrow{S} = (S_0, \ldots, S_d)$ and Éloïse moves to $(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{S})$.

Update of Λ. The memory Λ is updated after each visit to a vertex of the form $(p, a, \overrightarrow{R}, \theta)$. We have three cases depending on the kind of the last round:

- The round is a trivial bump and therefore a (q, a') transition was simulated. Let (p, ua) be the last vertex in Λ, then the updated memory is $\Lambda \cdot (q, ua')$.

- The round is a bump, and therefore a bump of colour i (where i is the colour of the round) starting with some transition $(q, a'b)$ and ending in a state $s \in S_i$ was simulated. Let (p, ua) be the last vertex in Λ. Then the memory becomes Λ extended by $(q, ua'b)$ followed by a sequence of moves, where Éloïse respects Φ, that ends by popping b and reach (s, ua') while having i as smallest colour. By definition of S_i such a sequence of moves always exists.
- The round is a stair and therefore we have simulated a $(q, a'b)$ transition. If (p, ua) denotes the last vertex in Λ, then the updated memory is $\Lambda \cdot (q, ua'b)$.

Therefore, with any partial play λ in $\widetilde{\mathbb{G}}$ in which Éloïse respects her strategy φ, is associated a partial play Λ in $\mathbb{G}_{\mathrm{abs}}$. An immediate induction shows that Éloïse respects Φ in Λ. The same arguments works for an infinite play λ, and the corresponding play Λ is therefore infinite, starts from $(p_{in}, \bot)$ and Éloïse respects Φ in that play. Therefore it is a winning play.

The following proposition is a direct consequence of how φ was defined.

Proposition 10. *Let λ be a partial play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot, (\varnothing, \dots, \varnothing), \Omega(p_{in}))$, ends in a vertex of the form $(p, a, \overrightarrow{R}, \theta)$, and where Éloïse respects φ. Let Λ be the play associated with λ built by the strategy φ. Then the following holds:*

1) *Λ ends in a vertex of the form (p, ua) for some $u \in A^*$.*
2) *θ is the smallest visited colour in Λ since a (or a symbol that was later rewritten as a) has been pushed.*
3) *Assume that Λ is extended, that Éloïse keeps respecting Φ and that the next move after (p, ua) is to some vertex (r, u). Then $r \in R_\theta$.*

Proposition 10 implies that the strategy φ is well defined when it provides a move to $\#$. Moreover, one can deduce that, if Éloïse respects φ, ff is never reached.

For plays that never reach the sink state $\#$, using the definitions of $\widetilde{\mathcal{G}}$ and φ, we easily deduce the following proposition.

Proposition 11. *Let λ be a play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot, (\varnothing, \dots, \varnothing), \Omega(p_{in}))$, and where Éloïse respects φ. Assume that λ never visit $\#$, let Λ be the associated play built by the strategy φ, and let $(\Lambda_i)_{i \geqslant 0}$ be its rounds factorisation. Let $(\lambda_i)_{i \geqslant 0}$ be the rounds factorisation of λ. Then, for every $i \geqslant 1$ the following hold:*

1) *λ_i is a bump if and only if Λ_i is a bump*
2) *λ_i has colour $mcol_i^{\Lambda}$.*

Now consider a play λ in $\widetilde{\mathbb{G}}$ starting from $(p_{in}, \bot, (\varnothing, \dots, \varnothing), \Omega(p_{in}))$ where Éloïse respects φ. Either the λ loops in $\#$ (hence is won by Éloïse). Or, thanks to Proposition 11 the sequence of visited colours in λ is $(mcol_i^{\Lambda})_{i \geqslant 0}$ for the corresponding play Λ in $\mathbb{G}_{\mathrm{abs}}$. Hence, using Proposition 9 we conclude that λ is winning if and only if Λ is winning; as Λ is winning for Éloïse, it follows that λ is also winning for her.

Converse implication

First note that in order to prove the converse implication one could follow the direct implication and consider the point of view of Abelard. Nevertheless the proof we give here starts from a winning strategy for Éloïse in $\widetilde{\mathbb{G}}$ and deduces a strategy for her in $\mathbb{G}_{\mathrm{abs}}$: this induces a more involved proof but has the advantage to lead to an effective construction of a winning strategy for Éloïse in $\mathbb{G}_{\mathrm{abs}}$ if one has an effective strategy for her in $\widetilde{\mathbb{G}}$.

Assume now that Éloïse has a winning strategy φ in $\widetilde{\mathbb{G}}$ from $(p_{in}, \bot, (\varnothing, \dots, \varnothing), \Omega(p_{\mathrm{in}}))$. Using φ, we build a strategy Φ for Éloïse in $\mathbb{G}_{\mathrm{abs}}$ for plays starting from $(p_{in}, \bot)$.

The strategy Φ maintains as a memory a partial play λ in $\widetilde{\mathbb{G}}$, that is an element in $\widetilde{V}^*$. At the beginning λ is initialised to $(p_{\mathrm{in}}, \bot, (\varnothing, \dots, \varnothing), \Omega(p_{\mathrm{in}}))$

For any play Λ where Éloïse respects Φ the following will hold.

- λ is a play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot, (\varnothing, \dots, \varnothing), \Omega(p_{\mathrm{in}}))$ and where Éloïse respects her winning strategy φ.
- The last vertex of λ is some $(p, a, \overrightarrow{R}, \theta)$ if and only if the current configuration in Λ is of the form (p, ua).
- If Éloïse keeps respecting Φ, and if a (or a symbol that rewrite it later) is eventually popped the configuration reached will be of the form (r, u) for some $r \in R_i$, where i is the smallest visited colour since a (or some symbol that was later rewritten as a) was on the stack.

Note that initially the previous invariants trivially hold.

In order to describe Φ, we assume that we are in some configuration (p, ua) and that the last vertex of λ is some $(p, a, \overrightarrow{R}, \theta)$. We first describe how Éloïse plays if $p \in Q_{\mathbf{E}}$, and then we explain how $\sqsubseteq$ is updated.

Choice of the move. Assume that $p \in Q_{\mathbf{E}}$. Then the move given by Φ depends on $\varphi(\lambda)$.

- If $\varphi(\lambda) = (q, a', \overrightarrow{R}, \min(\theta, \Omega(q)))$, Éloïse plays transition (q, a').
- If $\varphi(\lambda) = (p, a', \overrightarrow{R}, \theta, q, b)$, then Éloïse applies plays transition $(q, a'b)$.
- If $\varphi(\lambda) = \textit{tt}$, Éloïse plays transition (r, ε) for some state $r \in R_\theta$. Lemma 6 will prove that such an r always exists.

Update of λ. The memory λ is updated after each move (played by any of the two players). We have several cases depending on the last transition.

- If the last move was from (p, ua) to (q, ua') then the updated memory is $\lambda \cdot (q, a', \overrightarrow{R}, \min(\theta, \Omega(q)))$.
- If the last move was from (p, ua) to $(q, ua'b)$, let $(p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{S}) = \varphi(\lambda \cdot (p, a', \overrightarrow{R}, \theta, q, b))$. Intuitively, $\overrightarrow{S}$ describes which states Éloïse can force a play to reach if b is eventually popped. Then the updated memory is $\lambda \cdot (p, a', \overrightarrow{R}, \theta, q, b) \cdot (p, a', \overrightarrow{R}, \theta, q, b, \overrightarrow{S}) \cdot (q, b, \overrightarrow{S}, \Omega(q))$.
- If the last move was from (p, ua) to (r, u) the update of λ is as follows. One backtrack in λ until one finds a configuration of the form $(p', a', \overrightarrow{R'}, \theta', p'', a'', \overrightarrow{R})$ that is not immediately followed by a vertex of the form $(s, a'', \overrightarrow{R}, \theta'', i)$. This configuration is therefore in the stair that simulates the pushing of a'' onto the stack (here if $a'' \neq a$ then a'' was later rewritten as a). Call λ' the prefix of λ ending in this configuration. The updated memory is $\lambda' \cdot (r, a', \overrightarrow{R'}, \min(\theta', \theta, \Omega(r)), \theta) \cdot (r, a', \overrightarrow{R'}, \min(\theta', \theta, \Omega(r)))$. Formally, write $\lambda = \lambda_1 \odot \lambda_2 \odot \cdots \odot \lambda_k$ where $(\lambda_i)_{1 \leqslant i \leqslant k}$ is the round factorisation of λ. Let $h \leqslant k$ be the largest integer such that λ_h is a stair and let $\lambda_h = (p', a', \overrightarrow{R'}, \theta')(p', a', \overrightarrow{R'}, \theta', p'', a'')(p', a', \overrightarrow{R'}, \theta', p'', a'', \overrightarrow{R})(p'', a'', \overrightarrow{R}, \Omega(p''))$. Define $\lambda'_h = (p', a', \overrightarrow{R'}, \theta')(p', a', \overrightarrow{R'}, \theta', p'', a'')(p', a', \overrightarrow{R'}, \theta', p'', a'', \overrightarrow{R})(r, a', \overrightarrow{R'}, \min(\theta', \theta, \Omega(r)), \theta) \cdot (r, a', \overrightarrow{R'}, \min(\theta', \theta, \Omega(r)))$. Then the updated memory is $\lambda_1 \odot \lambda_2 \odot \cdots \odot \lambda_{h-1} \odot \lambda'_h$.

The following lemma gives the meaning of the information stored in $\sqsubseteq$.

Lemma 6. *Let Λ be a partial play in $\mathbb{G}_{\text{abs}}$, where Éloïse respects Φ, that starts from $(p_{in}, \bot)$ and that ends in a configuration (p, ua). We have the following facts:*

1) *The last vertex of λ is $(p, a, \overrightarrow{R}, \theta)$ with $\overrightarrow{R} \in (2^Q)^{d+1}$ and $0 \leqslant \theta \leqslant d$.*
2) *λ is a partial play in $\widetilde{\mathbb{G}}$ that starts from $(p_{in}, \bot, (\varnothing, \ldots, \varnothing), \Omega(p_{in}))$, that ends with $(p, a, \overrightarrow{R}, \theta)$ and where Éloïse respects φ.*
3) *θ is the smallest colour visited since a (or some symbol that was later rewritten as a) was pushed.*
4) *If Λ is extended by some move that pops a, the configuration (r, u) that is reached is such that $r \in R_\theta$.*

Proof: The proof goes by induction on Λ. We first show that the last point is a consequence of the second and third points. Assume that the next move after (p, ua) is to play a transition $(r, \varepsilon) \in \Delta(p, a)$. The second point implies that $(p, a, \overrightarrow{R}, \theta)$ is winning for Éloïse in $\widetilde{\mathbb{G}}$. If $p \in Q_{\mathbf{E}}$, by definition of Φ, there is some edge from that vertex to $\textit{tt}$, which means that $r \in R_\theta$ and allows us to conclude. If $p \in Q_{\mathbf{A}}$, note that there is no edge from $(p, a, \overrightarrow{R}, \theta)$ (winning position for Éloïse) to the losing vertex $\textit{ff}$. Hence we conclude the same way.

Let us now prove the other points. For this, assume that the result is proved for some play Λ, and let Λ' be an extension of Λ. We have two cases, depending on how Λ' extends Λ:

- Λ' is obtained by applying a transition of the form (q, a') or $(q, a'b)$. The result is trivial in that case.
- Λ' is obtained by applying a transition of the form (r, ε). Let (p, ua) be the last configuration in Λ, and let $\overrightarrow{R}$ be the last vector component in the last vertex of λ when in configuration (p, ua). By the induction hypothesis, it follows that $\Lambda' = \Lambda \cdot (r, u)$ with $r \in R_\theta$. Considering how λ is updated, and using the fourth point, we easily deduce that the new memory λ is as desired.

■

Actually, we easily deduce a more precise result.

Lemma 7. *Let Λ be a partial play in $\mathbb{G}_{\text{abs}}$ starting from $(p_{in}, \bot)$ and where Éloïse respects Φ and let $(\Lambda_i)_{i \geqslant 0}$ be its rounds factorisation. Let $(\lambda_i)_{i=0,\ldots,k}$ be the rounds factorisation of λ. Then the following holds:*

- *λ_i is a bump if and only if Λ_i is a bump.*
- *λ_i has colour $mcol_i^\Lambda$.*

Both lemmas 6 and 7 are for partial plays. A version for infinite plays would allow us to conclude. Let Λ be an infinite play in $\mathbb{G}_{\text{abs}}$. We define an infinite version of λ by considering the limit of the $(\lambda_i))_{i \geqslant 0}$ where λ_i is the memory after the i first moves in Λ. See Footnote 11 on page 36 for a similar construction. It is easily seen that such a limit always exists, is infinite and corresponds to a play won by Éloïse in $\widetilde{\mathbb{G}}$. Moreover the results of Lemma 7 apply.

Let Λ be a play in $\mathbb{G}_{\text{abs}}$ with initial vertex $(p_{\text{in}}, \bot)$, and where Éloïse respects Φ, and let λ be the associated infinite play in $\widetilde{\mathbb{G}}$. Therefore λ is won by Éloïse. Using Lemma 7 and Proposition 9, we conclude, as in the direct implication that Λ is winning.

Main Result

Theorem 10. *For any n-CPDP $\mathcal{A}_{\mathrm{lf}} = \langle \Gamma_{\mathrm{lf}}, Q_{\mathrm{lf}}, \Delta_{\mathrm{lf}}, q_{0,\mathrm{lf}} \rangle$ that* does not create n-links *and any associated parity game $\mathbb{G}_{\mathrm{lf}}$, one can construct an $(n-1)$-CPDP $\widetilde{\mathcal{A}} = \langle \widetilde{\Gamma}, \widetilde{Q}, \widetilde{\Delta}, \widetilde{q_0} \rangle$ and an associated parity game $\widetilde{\mathbb{G}}$ such that the following holds.*

- *$(q_{0,\mathrm{lf}}, \bot_n)$ is winning for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ if and only if $(\widetilde{q_0}, \bot_{n-1})$ is winning for Éloïse in $\widetilde{\mathbb{G}}$.*
- *If there is an $(n-1)$-CPDA transducer $\widetilde{\mathcal{S}}$ synchronised with $\widetilde{\mathcal{A}}$ realising a well-defined winning strategy for Éloïse in $\widetilde{\mathbb{G}}$ from $(\widetilde{q_0}, \bot_{n-1})$, then one can effectively construct an n-CPDA transducer $\mathcal{S}_{\mathrm{lf}}$ synchronised with $\mathcal{A}_{\mathrm{lf}}$ realising a well-defined winning strategy for Éloïse in $\mathbb{G}_{\mathrm{lf}}$ from the initial configuration $(q_{0,\mathrm{lf}}, \bot_n)$.*

Proof:

Following Example 23, $\mathcal{A}_{\mathrm{lf}}$ can be seen as an abstract pushdown automaton. In particular, we can apply the construction of Section D7. We claim that the resulting game $\widetilde{\mathbb{G}}$ is associated with an $(n-1)$-CPDP.

Indeed, one simply needs to consider how the graph $\widetilde{G}$ is defined and make the following observations concerning the local structure given in Figure 5 when $\mathbb{G}$ is played on the transition graph of an n-CPDP that does not create links.

1) For every vertex of the form $(p, a, \overrightarrow{R}, \theta)$, $(p, a, \overrightarrow{R}, \theta, q, b)$, $(p, a, \overrightarrow{R}, \theta, q, b, \overrightarrow{S})$ or $(s, a, \vec{R}, \theta', i)$, a and b are $(n-1)$-stack.
2) For every vertex of the form $(p, a, \overrightarrow{R}, \theta, q, b)$ or $(p, a, \overrightarrow{R}, \theta, q, b, \overrightarrow{S})$, one has $a = b$.

This implies that any vertex in $\widetilde{G}$ can be seen as a pair formed by a state in a finite set and an $(n-1)$-stack. Then one concludes the proof by checking that the edge relation is the one of an $(n-1)$-CPDP (for the transition to vertices $\sharp$ and ff one introduces vertices $(\sharp, a)$ and (ff, a) for any $(n-1)$-stack a).

Therefore, the first point follows by Theorem 9.

We now turn to the last point and therefore assume that there is an $(n-1)$-CPDA transducer $\widetilde{\mathcal{S}}$ synchronised with $\widetilde{\mathcal{A}}$ realising a well-defined winning strategy φ for Éloïse in $\widetilde{\mathbb{G}}$ from $(\widetilde{q_0}, \bot_{n-1})$. We argue that the strategy Φ constructed in the proof of Theorem 9 can be realised, when $\mathbb{G}_{\mathrm{abs}}$ is obtained from an n-CPDP $\mathcal{A}_{\mathrm{lf}}$ that does not create n-links, by an n-CPDA transducer $\mathcal{S}_{\mathrm{lf}}$ synchronised with $\mathcal{A}_{\mathrm{lf}}$.

For this, let us first have a closer look at Φ. The key ingredient in Φ is the play λ in $\widetilde{\mathbb{G}}$, and the value of Φ uniquely depends on $\varphi(\lambda)$. In particular, if φ is realised by an $(n-1)$-CPDA transducer $\widetilde{\mathcal{S}}$, it suffices to know the configuration of $\widetilde{\mathcal{S}}$ after reading λ in order to define Φ. We claim that it can be computed by an n-CPDA transducer $\mathcal{S}_{\mathrm{lf}}$ (synchronised with $\mathcal{A}_{\mathrm{lf}}$); the hard part being to establish that such a device can update correctly its memory.

Let $\lambda = v_0 v_1 \cdots v_\ell$ and let $r_\lambda = (p_0, s_0)(p_1, s_1) \cdots (p_\ell, s_\ell)$ be the run of $\mathcal{S}$ associated with λ, *i.e.* after having played $v_0 \cdots v_k$, $\mathcal{S}$ is in configuration (p_k, s_k). Denote by $Last(r_\lambda)$ the last configuration of r_λ, *i.e.* (p_ℓ, s_ℓ). To define Φ, $Last(r_\lambda)$ suffices but of course, in order to update $Last(r_\lambda)$, we need to recall some more configurations from r_λ. In the case where the last transition applies an order-k stack operation with $k < n$ (*i.e.* it is neither pop_n nor $push_n$), then the update is simple, as it consists in simulating one step of $\mathcal{S}$. If the last stack operation is $push_n$ then the update of λ consists in adding three vertices and the corresponding update of r_λ is simple (as the only operation on the $(n-1)$-stack is to rewrite the top_1-element). If the last stack operation is pop_n one needs to backtrack in λ (hence in r_λ): the backtrack is to some v_k with k maximal such that v_k is of the form $(p', a', \overrightarrow{R'}, \theta', p'', a'', \overrightarrow{R})$ and $v_{k+1} = (p'', a'', \overrightarrow{R}, \Omega(p''))$. Once v_k has been found, the update is fairly simple for both λ and r_λ (one simply extends the remaining prefix of λ by two extra vertices whose stack content is unchanged compared with the one in v_k).

Define the following set of indices where $\lambda = v_0 v_1 \cdots v_\ell$

$$Ext(\lambda) = \{h \mid v_h \text{ is of the form } (p', a', \overrightarrow{R'}, \theta', p'', a'', \overrightarrow{R}) \text{ and } v_{h+1} = (p'', a'', \overrightarrow{R}, \Omega(p''))\} \cup \{\ell\}$$

Note that after a partial play Λ the cardinality of $Ext(\lambda)$ is equal to the height of the stack in the last configuration of Λ.

For any partial play Λ in $\mathbb{G}_{\mathrm{lf}}$ define the following n-stack (note that it does not contain any n-link)

$$Mem(\Lambda) = [\, s'_{k_1} s'_{k_2} \cdots s'_{k_h} \,]$$

where we let

- $Ext(\lambda) = \{k_1 < \cdots k_h\}$, λ being the memory associated with Λ as in the proof of Theorem 9
- s'_j is the $(n-1)$-stack obtained from s_j (recall that (p_j, s_j) denotes the j-th configuration of r_λ) by appending p_j to its top_1-symbol (*i.e.* we work on an enriched stack alphabet).

Note that $Last(r_\lambda)$ is essentially $top_1(Mem(\Lambda))$ as the only difference is that now the control state is stored in the stack. Moreover $Mem(\Lambda)$ can easily be updated by an n-CPDA transducer: for the case of a transition involving an order-k stack operation with $k < n$ one simulates $\mathcal{S}$ on $top_1(Mem(\Lambda))$; for the case of a transition involving a $push_n$ one first

simulates $\mathcal{S}$ on $top_1(Mem(\Lambda))$ (as one may do a rew_1 before $push_n$) and then makes a $push_n$ to duplicate the topmost $(n-1)$-stack in $Mem(\Lambda)$; finally, for the case of a pop_n, one simply needs to do a pop_n in $Mem(\Lambda)$ to backtrack and then update the control state. This is how we define $\mathcal{S}_{\mathrm{lf}}$[12].

The fact that $\mathcal{S}_{\mathrm{lf}}$ is synchronised with $\mathcal{A}_{\mathrm{lf}}$ comes from the definition of how $\mathcal{S}_{\mathrm{lf}}$ behaves when the transition in $\mathcal{A}_{\mathrm{lf}}$ involves a pop_n or a $push_n$, and for the other cases it follows from the initial assumption of $\mathcal{S}$ being synchronised with $\widetilde{\mathcal{A}}$. ∎

If we summarise, the overall blowup in the transformation from $\mathbb{G}_{\mathrm{lf}}$ to $\widetilde{\mathbb{G}}$ given by Theorem 10 is as follows.

Proposition 12. *Let $\mathcal{A}_{\mathrm{lf}}$ and $\widetilde{\mathcal{A}}$ be as in Theorem 10. Then the set of states of $\widetilde{\mathcal{A}}$ has size $\mathcal{O}(2^{2(d+1)|Q_{\mathrm{lf}}|})$ and the stack alphabet of $\widetilde{\mathcal{A}}$ has size $\mathcal{O}(|\Gamma_{\mathrm{lf}}|)$.*

Finally the set of colours used in $\mathbb{G}_{\mathrm{lf}}$ and $\widetilde{\mathbb{G}}$ are the same.

Proof: By construction. ∎

8) Proof of Theorem 5:

The proof of Theorem 5 consists in combining theorems 6, 7 and 10. Indeed, starting from an n-CPDP, by $(n-1)$ successive applications of this three results, we obtain a 1-CPDP parity game. If we apply to this latter (pushdown) game the construction of Section D7 we end up with a game on a finite graph. Solving this game and following the chain of equivalences provided by theorems 6, 7 and 10 concludes the proof.

Concerning complexity, one step of successive application of the construction in theorems 6, 7 and 10 results in an $(n-1)$-CPDP with stack alphabet of size $\mathcal{O}(|\Gamma|^2 \cdot 2^{|Q|(d+1)^{n+5}})$ and state set of size $\mathcal{O}(2^{|Q|(d+1)^{n+5}})$ (complexity follows from propositions 5, 8 and 12.

If one let $\exp_k$ be the function defined by $\exp_0(x) = x$ for all x and $\exp_{k+1}(x) = 2^{\exp_k(x)}$, we conclude that the 1-CPDP obtained after $(n-1)$ successive applications of the three reductions has a stack alphabet of size $\mathcal{O}(|\Gamma|^{2(n-1)} \cdot \exp_{n-1}(|Q|(d+1)^{n+5}))$ and state set of size $\mathcal{O}(\exp_{n-1}(|Q|(d+1)^{n+5}))$. Finally the finite game we obtain is a parity game with $(d+1)$ colours on a graph with $\mathcal{O}(|\Gamma|^{4(n-1)}\exp_n(|Q|(d+1)^{n+6}))$ vertices. This latter game can be solve in $\mathcal{O}([|\Gamma|^{4(n-1)}\exp_n(|Q|(d+1)^{n+6})]^d)$

In particular the overall complexity of deciding the winner in an n-CPDP parity game is:

- n-times exponential in the number of states of the CPDP;
- $(n+1)$-times exponential in the number of colours;
- polynomial in the stack alphabet of the CPDP.

Hardness already holds when one considers reachability condition (*i.e.* does the play visit a configuration with a final control state?) for games generated by higher-order pushdown automata (*i.e.* CPDP that never use $collapse$). A self content proof of this result was established by Thierry Cachat and Igor Walukiewicz, but was unfortunately not published [A1].

[12]Technically speaking, if we impose that a transition of $\mathcal{S}_{\mathrm{lf}}$ does a rew_1 (or id) followed by another stack operation, we may not be able to do the update of the stack after doing a pop_n. However, we can use the same trick as the one used to define $\mathcal{A}_{\mathrm{rk}}$ (see Remark 8).

E. Proofs Omitted in Section V

1) Proof of Theorem 3:

Theorem 3. *Labeled recursion schemes as well as CPDA have the effective MSO selection property.*

Proof: Let $\varphi(X_1, \cdots, X_\ell)$ be a monadic second order formula with ℓ second-order free variables, and let $\mathcal{S} = \langle\, \Sigma, N, \mathcal{R}, Z, \bot \,\rangle$ be a labelled recursion scheme.

Relying on Theorem 1, we consider a CPDA $\mathcal{A}$ such that $\mathrm{Tree}^{\bot}(\mathcal{S}) = \mathrm{Tree}^{\bot}(\mathcal{A})$.

Let t_φ be a term over the ranked alphabet $\Sigma \times \{0,1\}^\ell$. We say that t_φ is a *marking* of a term t over the ranked alphabet Σ iff t is obtained from t_φ by forgetting the $\{0,1\}^\ell$ component. Formally, if we let π denotes the natural projection from $\Sigma \times \{0,1\}^\ell$ into Σ, we require that $t = \{\pi(u_\varphi) \mid u_\varphi \in t_\varphi\}$ and that for all $u_\varphi, v_\varphi \in t_\varphi$, $\pi(u_\varphi) = \pi(v_\varphi) \Rightarrow u_\varphi = v_\varphi$ (*i.e.* π is injective on t_φ).

Let t_φ be a marking of a tree t. Then we define for any $1 \leqslant i \leqslant \ell$ the set $U_i = \{\pi(u_\varphi) \mid u_\varphi$ ends by some $(a_u, b_{u,1}, \ldots, b_{u,\ell})$ with $b_{u,i} = 1\}$ of nodes in t which are the image by π of a node whose i-th component is 1.

Thanks to the well-known equivalence between logic and tree automata, there is a nondeterministic parity tree automaton $\mathcal{B}_\varphi$ working on $\Sigma \times \{0,1\}^\ell$ trees such that a tree t_φ is accepted by $\mathcal{B}_\varphi$ iff t_φ is the marking of a tree t such that $\varphi(X_1 \leftarrow U_1, \ldots X_\ell \leftarrow U_\ell)$ holds in t.

Recall that acceptance of a tree by a nondeterministic parity tree automaton can be seen as existence of a winning strategy in a parity game that is (informally) played as follows. The two players, Éloïse and Abelard move down the tree a pebble to which is attached a state of the automaton; the play starts at the root (with initial state attached to the pebble); at each round Éloïse provides a valid transition (w.r.t the current state and the current node label) of the automaton and Abelard moves the pebble to some son and update the state attached to the pebble according to the transition chosen by Éloïse. In case the pebble reach a leaf, the play ends and Éloïse wins iff the state is final (we have final states in the tree automaton to handle finite branches); otherwise the play is infinite and Éloïse wins iff the smallest infinitely visited priority is even.

For some t_φ, the previous acceptance game is easily seen to be a collapsible pushdown games. The underlying arena is essentially a synchronised product of the transition graph of a collapsible pushdown process with the finite graph corresponding to $\mathcal{B}_\varphi$. Now consider a variant of this game where instead of checking whether a given t_φ is accepted by $\mathcal{B}_\varphi$ the players wants to check, for a given tree t, whether there exists some t_φ such that t_φ is accepted by $\mathcal{B}_\varphi$ and t_φ is a marking of t. The game is essentially the same, except that now Éloïse is also giving the marking of the current vertex (*i.e.* π^{-1}). Again, this leads to a collapsible pushdown game and one directly checks that Éloïse wins from the root iff there is a marking of t that is accepted by $\mathcal{B}_\varphi$. Call $\mathbb{G}$ this game and call $\mathcal{A}'$ the underlying CPDP.

Apply Theorem 5 to $\mathbb{G}$. Then either Éloïse has no winning strategy from the initial configuration (call it $(q_0, [\ldots[\,\bot\,]_1 \ldots]_n)$) and we are done (there is no selector). Otherwise one can effectively construct an n-CPDA transducer $\mathcal{T}$ *synchronised* with $\mathcal{A}'$ realising a well-defined *winning* strategy for Éloïse in $\mathbb{G}$ from $(q_0, [\ldots[\,\bot\,]_1 \ldots]_n)$. As $\mathcal{A}'$ and $\mathcal{T}$ are synchronised, we can consider their synchronised product, call it $\mathcal{A}''$. Hence in $\mathcal{A}''$ the configurations contain extra informations (coming from $\mathcal{T}$); in particular, for any configuration, if the control state from the $\mathcal{A}'$ component is controlled by Éloïse, then the control state from the $\mathcal{T}$ component provides the next move Éloïse should play: in particular, it provides a transition of the tree automaton, together with information regarding the marking. Transform $\mathcal{A}''$ by removing every transition that is not consistent with the strategy described by $\mathcal{T}$: then the tree generated by this new CPDA is isomorphic to some t_φ (that is a marking of t) together with an accepting run of $\mathcal{B}_\varphi$ on it. Now if we forget the component from $\mathcal{B}_\varphi$ we obtain a CPDA $\mathcal{A}_\varphi$ that generates a marking t_φ of t.

Finally, as we can transform $\mathcal{A}_\varphi$ back to a labeled recursion scheme, we get $\mathcal{S}_\varphi$ as expected.

The proof for CPDA follows the same line, except that one directly work on CPDP games.

■

2) Proof of Corollary 3:

Corollary 3. *The μ-calculus model-checking of trees generated by recursion schemes is* polynomial *under the assumption that the arity of types and the formula are bounded above by a constant.*

Proof: The μ-calculus model-checking of trees generated by recursion schemes reduces to solving CPDP parity games. If the arity of types and the formula are bounded above by a constant, the number of states in the CPDP generating the arena as well as the number of colours in the game are bounded as well (see Theorem 1). Then, thanks to the complexity analysis (see section D8) of Theorem 5, we easily conclude. ■

APPENDIX BIBLIOGRAPHY

[A1] T. Cachat and I. Walukiewicz. The complexity of games on higher order pushdown automata. *CoRR*, abs/0705.0262, 2007.

[A2] B. Courcelle. A representation of trees by languages I. *Theoret. Comput. Sci.*, 6:255–279, 1978.

[A3] B. Courcelle. A representation of trees by languages II. *Theoret. Comput. Sci.*, 7:25–55, 1978.

[A4] T. Knapik, D. Niwiński, P. Urzyczyn, and I. Walukiewicz. Unsafe grammars and panic automata. In *Proc. of ICALP'05*, volume 3580 of *LNCS*, pages 1450–1461. Springer, 2005.

[A5] D. A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.